Demoiselle Nimble

Guia do Usuário

Emerson Saito

Rodrigo Hjort

Serge Rehem

Guia do usuário

Apresentamos o guia do usuário do Demoiselle Nimble, que é uma ferramenta de apoio ao Desenvolvimento do *Demoiselle Framework* a partir da versão *2.0*.

Importante

O Demoiselle Nimble está disponível para o usuário final nas seguintes opções:

- Desktop
- Eclipse Plugin
- NetBeans Plugin

Sobre Demoiselle Nimble

Informações básicas sobre a ferramenta

1.1. O que é

O *Demoiselle Nimble* é um processador automatizado de templates [1], sendo estes genéricos e podendo inclusive ser criados pelo próprio utilizador.

Podemos considerá-lo como uma evolução da ferramenta atual o *Demoiselle Wizard Eclipse Plugin* [http://demoiselle.sourceforge.net/wizard/] que como o próprio nome já diz é um especificamente um plugin para a IDE Eclipse

Nota

Um *processador de templates* (i.e., template processor, também conhecido como template engine ou template parser) é um software ou um componente de software projetado para combinar um ou mais templates com um modelo de dados a fim de produzir um ou mais documentos resultantes, podendo estes serem páginas web, arquivos de texto ou códigos fontes.

No caso específico do Demoiselle Nimble, no template são definidas estruturas de arquivos e scripts de transformação utilizando linguagens como Velocity e Groovy. Uma vez iniciado o processamento, este faz uso de variáveis cujos valores foram definidos pelo usuário e que permitem com que os diretórios e arquivos resultantes possam ser dinamicamente criados e ou modificados.

1.2. Templates Disponíveis

São disponibilizados com o Demoiselle Nimble os seguintes templates voltados para a geração automática de código para o Framework Demoiselle:

- *Demoiselle v1 Create JSF Application:* cria a estrutura básica de uma aplicação baseada no Demoiselle Framework Versão 1.x
- *Demoiselle v1 Generate CRUD:* cria os artefatos para as funcionalidades de CRUD (Create, Read, Update and Delete) de uma entidade no Demoiselle Framework Versão 1.x
- *Demoiselle V2 Create JSF Application:* cria uma aplicação completa e funcional baseada no Demoiselle Framework versão 2.x com JSF e JPA (usa internamente um arquétipo Maven da versão mais recente do Demoiselle)
- *Demoiselle v2 Create VAADIN Application:* cria uma aplicação baseada na extensão Demoiselle Vaadin (com Demoiselle Framework versão 2.x)
- *Demoiselle v2 Generate JSF-PrimeFaces2 CRUD:* cria todos os artefatos necessários para as funcionalidades CRUD (Create, Read, Update and Delete) de uma dada entidade baseando-se no Demoiselle Framework versão 2.x, e utilizando a biblioteca JSF2 PrimeFaces versão 2.x
- *Demoiselle v2 Generate JSF2 (PrimeFaces3 Mobile) CRUD:* cria todos os artefatos necessários para o CRUD (Create, Read, Update and Delete) de uma dada entidade baseando-se no Demoiselle Framework versão 2.x, e utilizando a biblioteca JSF2, especializada para WEB Mobile, PrimeFaces-Mobile versão 3.x

---

[1] Para informações gerais sobre utilitários processadores de template, acesse o site *http://en.wikipedia.org/wiki/Template_processor*

- *Demoiselle v2 Generate JSF2 (PrimeFaces 3) CRUD:* cria todos os artefatos necessários para o CRUD (Create, Read, Update and Delete) de uma dada entidade baseando-se no Demoiselle Framework versão 2.x, e utilizando a biblioteca JSF2 PrimeFaces versão 3.x
- *Demoiselle v2 Generate Vaadin CRUD:* cria os artefatos para o CRUD de uma entidade na extensão Demoiselle Vaadin (com Demoiselle Framework versão 2.x)

Instalação

Orientações de instalação e configuração para o modo Desktop

Atualmente há dois modos para fazer a instalação do Demoiselle Nimble, que são:

2.1. Via pacotes Debian GNU/Linux

Foi disponibilizado um pacote Debian chamado demoiselle-nimble para o Demoiselle Nimble no repositório do projeto Demoiselle Infra. Para instalá-lo nesta plataforma, basta executar o comando a seguir em um terminal do Linux:

```
# apt-get install demoiselle-nimble
```

Para maiores informações sobre a configuração e o modo de utilização dos pacotes de software do projeto Demoiselle Infra para Linux, acesse o site *http://demoiselle.sourceforge.net/infra/*

Nota

Recomendamos utilizar a solução Demoiselle Infra, que irá resolver as outras questões de ambiente de desenvolvimento para o Demoiselle

2.2. Instalação a partir do download

- Pré-requisitos

  Antes de começar a usar o Demoiselle Nimble, você precisa ter instalado ou instalar uma *JDK* - Java Development Kit. Nos sistemas operacionais linux recomendamos a OpenJDK

  Nos sistemas operacionais proprietários normalmente está disponível a implementação da Oracle: *Site de Download da SUN-JDK* [http://www.oracle.com/technetwork/java/javase/downloads/index.html]

  Veja que não basta ter simplesmente uma JRE - Java Runtime Environment, é necessária uma JDK pois o Nimble executa operações que necessitam de bibliotecas que não estão disponíveis numa JRE. Além disso, é preciso definir a variável de ambiente JAVA_HOME apontando para o caminho onde se localiza a JDK (ex: /usr/lib/jvm/java-6-openjdk/).

  ### Exemplo SO LINUX.

  ```
  export JAVA_HOME=/usr/lib/jvm/java-6-openjdk/
  ```

  No LINUX, é preciso incluir no arquivo /etc/profile

  ### Exemplo SO ms-windows.

  ```
  set JAVA_HOME=C:\Arquivos de programas\Java\jdk1.6.0_17\
  ```

Para ms-windows, o indicado é abrir as configurações de ambiente (Icone Meu Computador) e incluir uma nova variável de ambiente.

Nota

Recomenda-se utilizar a versão 1.5 ou superior da JDK. Porém se seu objetivo é utilizar a versão 2.0 do Demoiselle a versão mínima recomendada é a 1.6

Para utilizar alguns dos templates de criação de aplicações (create-app-*), também é preciso ter instalado e configurado o *APACHE MAVEN* [http://maven.apache.org]

No ambiente LINUX Ubuntu/Debian o pacote Demoiselle Infra também já fornece um pacote para esta ferramenta:

```
# apt-get install demoiselle-maven2
```

Na página de *Download* [http://maven.apache.org/download.html] há uma seção chamada *Installation Instructions* que orienta a instalação e configuração desta ferramenta.

- Passo a passo
  - Faça o *Download* [http://sourceforge.net/projects/demoiselle/files/tools/nimble] da última release do Demoiselle Nimble (disponível nos formatos ZIP e TGZ)
  - Extraia os arquivos em uma localização conveniente (ex: C:\Demoiselle no Windows ou ~/demoiselle no Linux)
  - Crie uma variável chamada DEMOISELLE_HOME que aponte para o caminho definido no passo anterior

Exemplo SO LINUX.

```
export DEMOISELLE_HOME=~/demoiselle

export PATH=$PATH:$DEMOISELLE_HOME/bin
```

No LINUX, é preciso incluir no arquivo /etc/profile Altere os direitos dos arquivos em "/bin" para permissão de execução `chmod +x $DEMOISELLE_HOME/bin/demoiselle`

Exemplo SO ms-windows.

```
set DEMOISELLE_HOME=C:\Demoiselle

set DEMOISELLE=%DEMOISELLE_HOME%\bin

set PATH=%PATH%;%DEMOISELLE%
```

Para ms-windows, o indicado é abrir as configurações de ambiente (Icone Meu Computador) e incluir novas variáveis de ambiente.

  - Abra um *prompt* de comando (no Windows) ou um terminal (no Linux) e digite o comando *"demoiselle"* a fim de verificar o sucesso da instalação

Guia Rápido

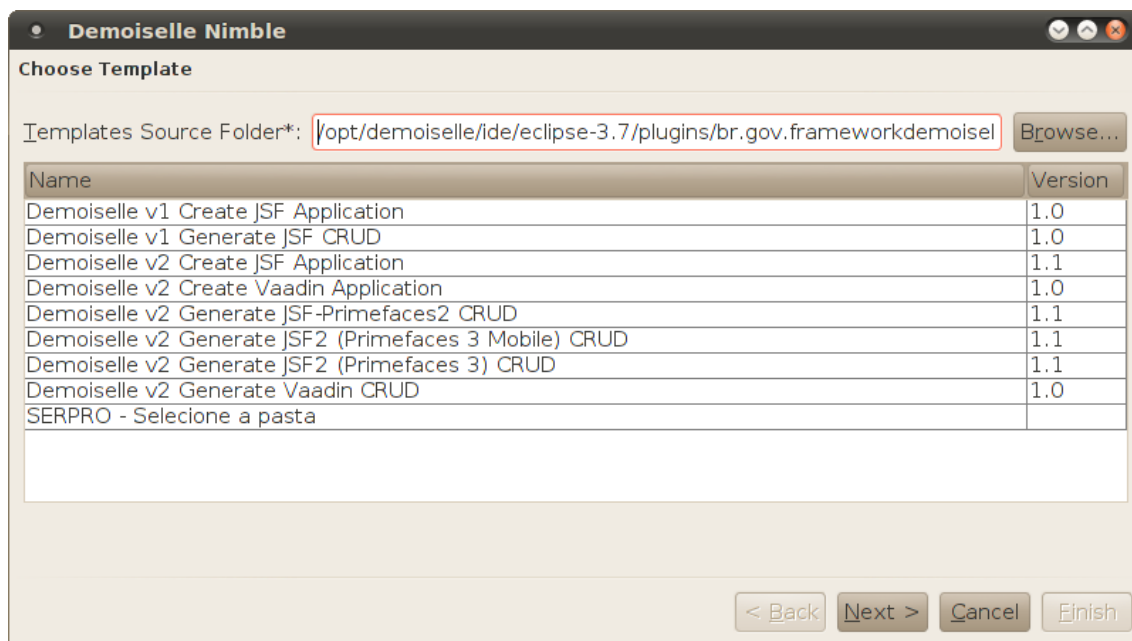
Guia de uso para o modo *Desktop*

Após a instalação e configuração do *Demoiselle Nimble* no ambiente Desktop (Linux ou ms-windows) a ferramenta estará pronta para execução. As formas de execução estão dividas em 2(dois) modos:

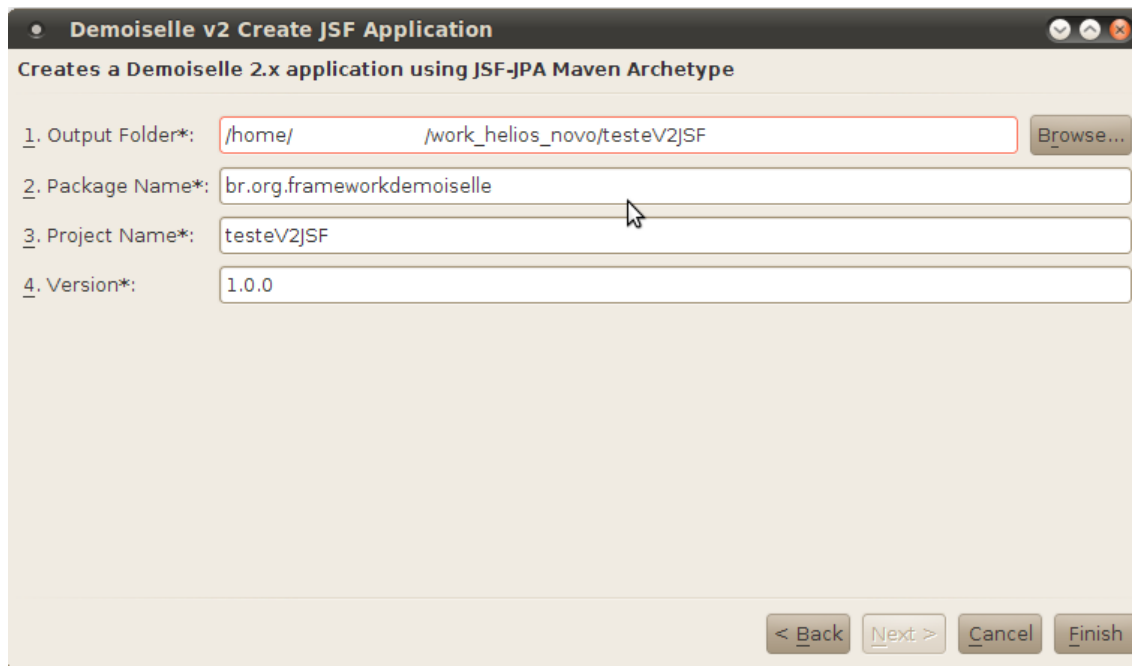
3.1. Interface Gráfica

O Demoiselle Nimble apresenta uma opção de interface gráfica do tipo *JAVA-SE SWING* [http://java.sun.com/docs/books/tutorial/uiswing/] e seu acionamento é simples, basta abrir um *prompt* de comando (no Windows) ou um terminal (no Linux) e digite o comando:

```
demoiselle -g
```

Para usuários do ms-windows acostumados a usar o *windows Explorer* para encontrar os arquivos instalados no computador, basta também procurar o arquivo *demoiselle.bat* localizado na pasta /bin do diretório onde foi "instalado" o Demoiselle Nimble. Ao acionar este arquivo desta maneira o *"defaul"* é a interface gráfica.

Interface Gráfica do Demoiselle Nimble. Tela Principal

Nesta tela Inicial há o botão *Browse...* que permite que seja selecionado o diretório onde estão os *Templates* a serem executados. Com a instalação do Demoiselle Nimble o diretório */templates* é criado com os *"defaults"* disponibilizados pelo Demoiselle. Lembrando que também é possível serem criados novos.

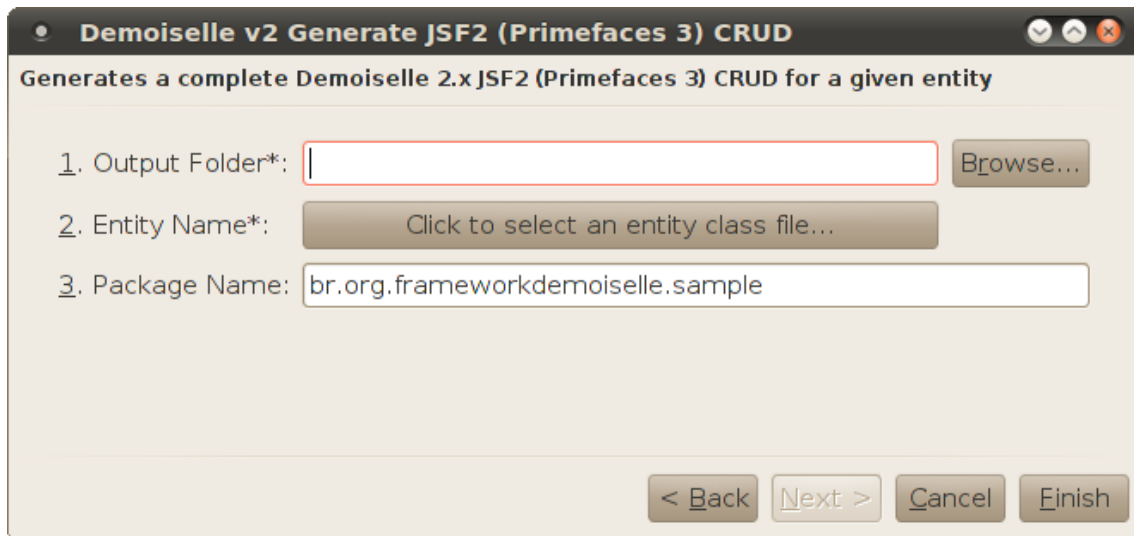
Nesta tela, além da possibilidade de escolha dos diretórios de templates, deve ser selecionado o *Template* que deseja executar. E clicar no botão *Next* > para prossegir a execução.

As funções dos demais botões, na parte inferior da tela, são:

- <*Back*: Retorna à tela anterior (Desabilitado na primeira tela)

- *Next* >: Segue para a ação seguinte (próxima tela)
- *Cancel*: Cancela a operação e fecha a interface gráfica.
- *Finish* Aciona a execução do Template. Só é habilitado quando todas as variáveis (informações) necessárias foram completadas

Para os *Templates* fornecidos pelo Demoiselle, conforme forem escolhidos na tela principal, serão apresentadas as seguintes telas:

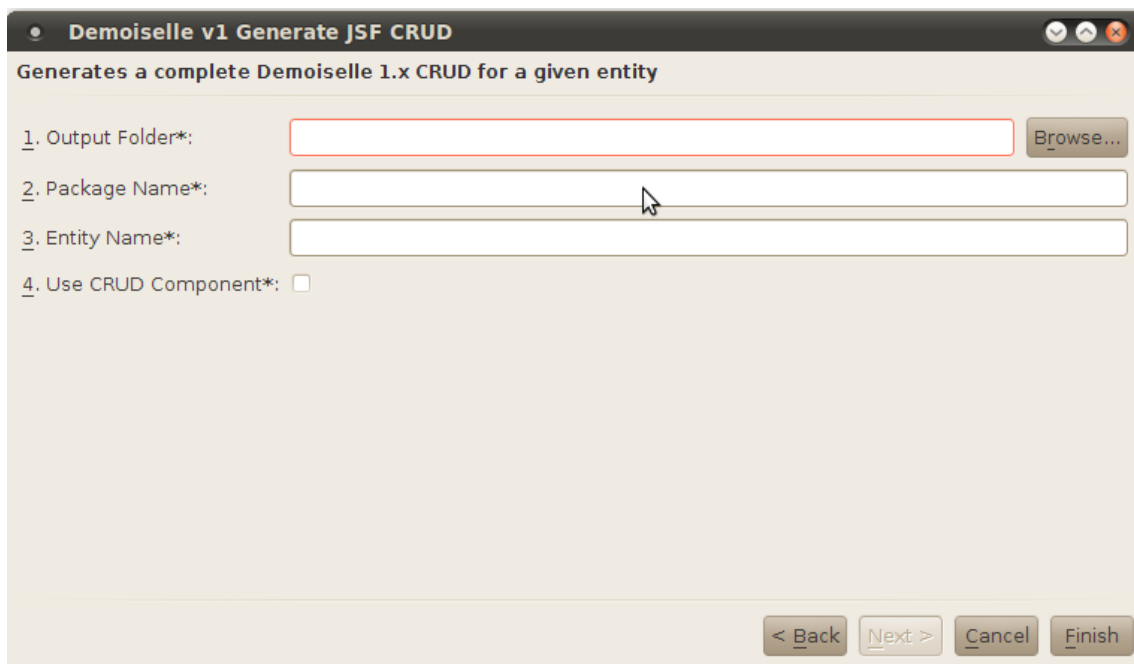
3.1.1. Templates de Criação de Aplicações

Tela dos Templates de Criação de Aplicação

As informações solicitadas nas telas são as mesmas para criação de aplicações JSF com Demoiselle v1, JSF com Demoiselle v2 e Vaadin com Demoiselle v2, pois o *Template* faz somente a chamada ao Maven e aciona o arquétipo Demoiselle já disponível no repositório no SourceForge

Sendo elas:

- *1.Output Folder*: Informar ou selecionar (usando botão "Browse...") em qual diretório deseja que o projeto seja criado. A dica aqui é usar o diretório do *workspace*, se deseja usar o Eclipse como IDE de Desenvolvimento
- *2.Package Name*: Informar o nome do pacote JAVA padrão da sua aplicação (ex: br.org.frameworkdemoiselle.)
- *3.Project Name*: O nome da aplicação/Projeto a ser criada.
- *4.Version* O número da versão inicial.

3.1.2. Templates de Geração de CRUD para JSF e VAADIN com Demoiselle v2.x

Um CRUD acrônimo de Create, Retrieve, Update e Delete em língua Inglesa, são as operações básicas relacionadas ao uso de bancos de dados em aplicações, assim nestes Templates, os produtos gerados são as classes para implementação destas operações, e também todos os artefatos de configuração e interface (páginas XHTML).

Tela dos Templates de Geração de CRUD para Aplicações JSF e VAADIN com Demoiselle v2.x

São estas as informações necessárias para o processamento destes templates

- *1.Output Folder*: Informar ou selecionar (usando botão "Browse...") em qual diretório deseja que o projeto seja criado. A dica aqui é usar o diretório do *workspace*, se deseja usar o Eclipse como IDE de Desenvolvimento
- *2.Package Name*: Informar o nome do pacote JAVA padrão da sua aplicação (ex: br.org.frameworkdemoiselle.)
- *3.Entity Name*: O nome da Entidade para qual será gerado o CRUD

3.1.3. Template de Geração de CRUD para JSF com Demoiselle v1.x

Para versão 1.x do Demoiselle Framework a apresentação da tela apresenta uma pequena diferença

Tela do Templates de Geração de CRUD para Aplicações JSF com Demoiselle v1.x

São estas as informações necessárias para o processamento deste template

- *1.Output Folder*: Informar ou selecionar (usando botão "Browse...") em qual diretório deseja que o projeto seja criado. A dica aqui é usar o diretório do *workspace*, se deseja usar o Eclipse como IDE de Desenvolvimento
- *2.Package Name*: Informar o nome do pacote JAVA padrão da sua aplicação (ex: br.org.frameworkdemoiselle.)
- *3.Entity Name*: O nome da Entidade para qual será gerado o CRUD
- *4.Use CRUD component*: Esta opção deve ser marcada quando a aplicação alvo utiliza o componente Demoiselle-CRUD.

3.2. Linhas de comando

A outra opção para uso do Demoiselle Nimble para *Desktop* é através de linhas de comando. E neste caso haverão algumas diferença conforme o Sistema Operacional que estiver sendo utilizado. Assim neste guia dividermos os exemplos seções para cada um.

Um dos motivos para se usar linha de comando é a possibilidade passagem de parâmetros para o aplicativo, os parâmetros do Demoiselle Nimble são estes:

- **-h**: apresenta um texto de ajuda.
- **-g**: aciona a inteface gráfica.
- **-i**: informa o diretório de templates
- **-o**: especifica o diretório de saida, onde serão gerados os artefatos
- **<<Nome do Template>>**: Nome do template a ser processado
- **-v []lista de variáveis=valor**: lista com variáveis que serão usadas pelo template

3.2.1. Executando em ambiente LINUX

Boa parte dos usuários LINUX estão habituados a utilizar os terminais de linhas de comando para executar operações com aplicativos, assim o Demoiselle Nimble também oferece esta opção.

Tendo sido configuradas corretamente as variáveis de ambiente, as seguintes operações podem ser executadas:

- demoiselle <<Nome do Template>>

  Exemplo

  ```
  demoiselle create-app-jsf
  ```

- demoiselle <<Nome do Template>> <<Parâmetros>>

  Exemplo

  ```
  demoiselle create-app-jsf -i /templates/ -o /temp/
  ```

```
demoiselle create-app-jsf -i /templates/ -o /temp/
br.gov.frameworkdemoiselle.sample myApp 1.0.0
```

- Também é possível passar parâmetros para a interface Gráfica

  Exemplo

```
demoiselle create-app-jsf -g -i ../templates/ -o /temp/
```

```
demoiselle create-app-jsf -g -i ../templates/ -o /temp/
br.gov.frameworkdemoiselle.sample myApp 1.0.0
```

3.2.2. Executando em ambiente MS-Windows [1]

Embora os usuário de sistemas operacionais MS-windows não costumem utilizar terminais de linhas de comandos, o Demoiselle-Nimble também oferece este opção. Atualmente com algumas diferenças no uso, por ser a primeira versão do *script (.bat)*

Novamente, é preciso que tenham sido configuradas corretamente as variáveis de ambiente, e abrir um *"Prompt de Comando"* para que as seguintes operações possam ser executadas:

- demoiselle <<Parâmetros>>

  Exemplo

```
demoiselle -i c:\temp\templates\create-app-jsf -o c:\temp\
```

```
demoiselle -i c:\temp\templates\crate-app-jsf -o c:\temp -v
projectName=ProjetoNovo
```

- Também é possível passar parâmetros para a interface Gráfica

  Exemplo

```
demoiselle -g -i c:\temp\templates\create-app-jsf -o c:\temp
```

[1] MS-Windos é marca registrada da Microsoft

Plugin para Eclipse IDE

Guia de uso para o *Puglin para Eclipse*

Uma das opções de uso do Demoiselle Nimble é a sua integração à *IDE Eclipse* [http://www.eclipse.org/downloads/] através de um plugin.

A vantagem no uso deste plugin é que com a sua integração com a IDE, alguns parâmetros como diretórios de *Templates*, diretório de saída, nome do projeto, podem ser fornecidos automaticamente pela própria IDE.

Como o Demoiselle-Nimble foi concebido para ser independente de Plataforma, o Eclipse faz apenas a chamada à interface gráfica do Demoiselle-Nimble passando os parâmetros necessários e as atulizações no ambiente (workspace).

Assim, as instruções de uso dos *templates* são as mesmas contidas no *Guia - Interface Gráfica*

Neste capítulo vamos nos ater apenas os detalhes referentes à intalação e uso no Eclipse

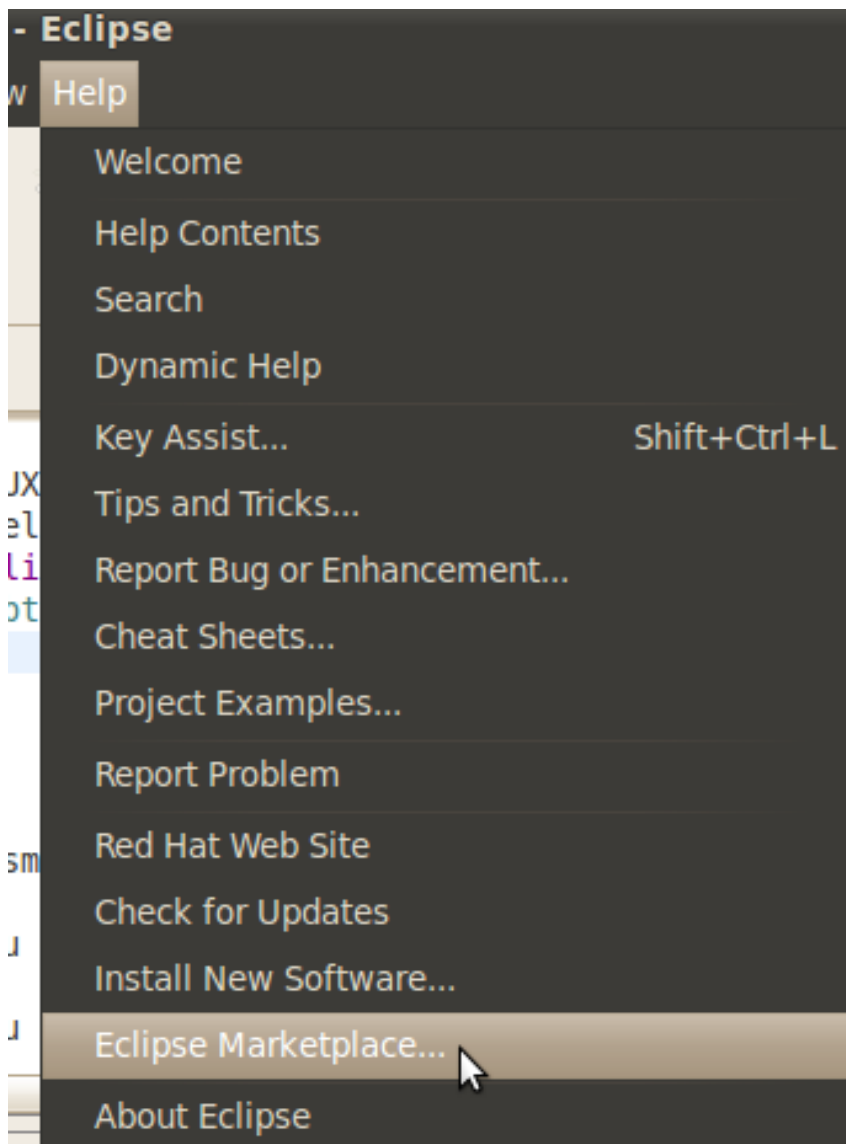
4.1. Instalação

Aqui, lembramos novamente que para usuários LINUX/DEBIAN existe o projeto Infra *http://demoiselle.sourceforge.net/infra/* que facilita a instalação das ferramentas que o Demoiselle recomenda para uso em seu desenvolvimento. Usando o pacote mais atual não é necessário fazer a instalação. Neste caso é só seguir para o item: *Seção 4.2, “Instruções de Uso”*

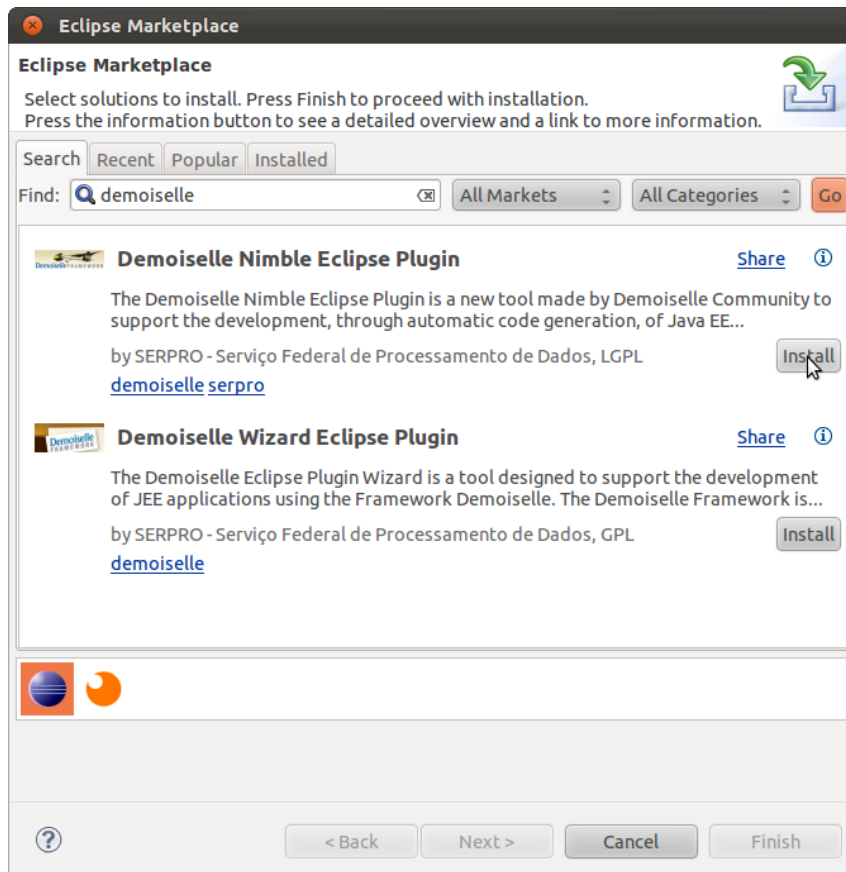
4.1.1. Instalação no Eclipse Indigo ou superior

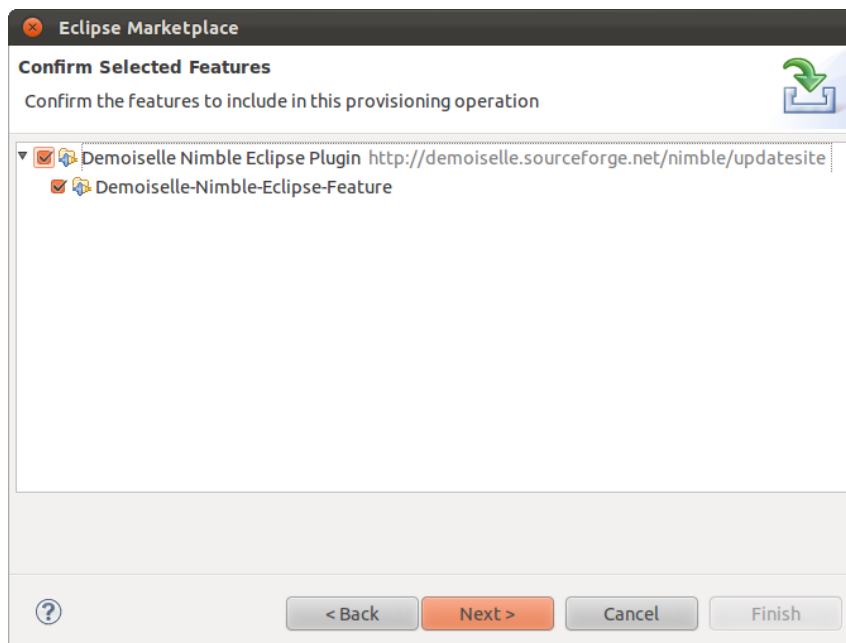
Nas versões mais atuais existe um serviço chamado Eclipse MarketPlace, que facilita bastante o processo de instalação de plugins.

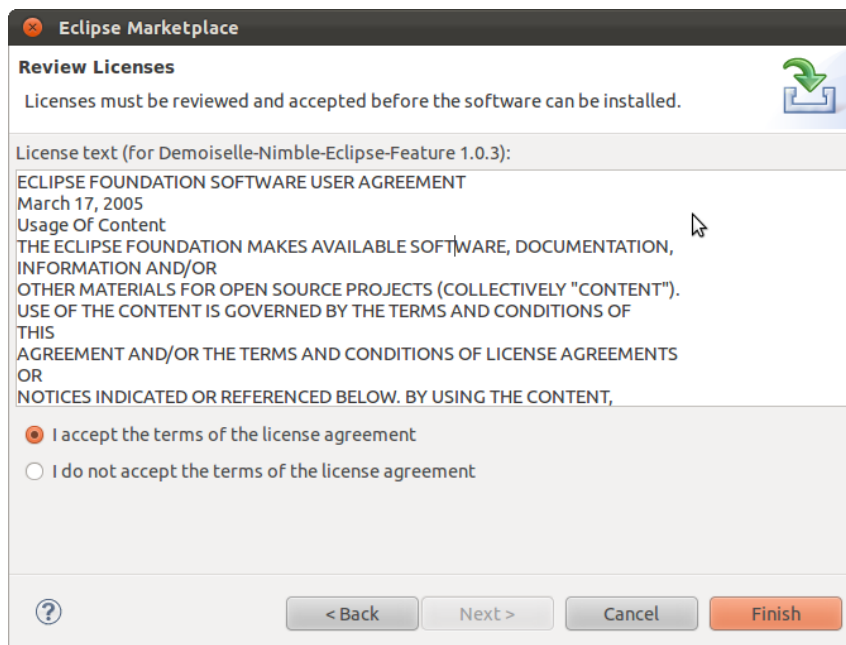
O primeiro passo é encontrar esse item no Eclipse.

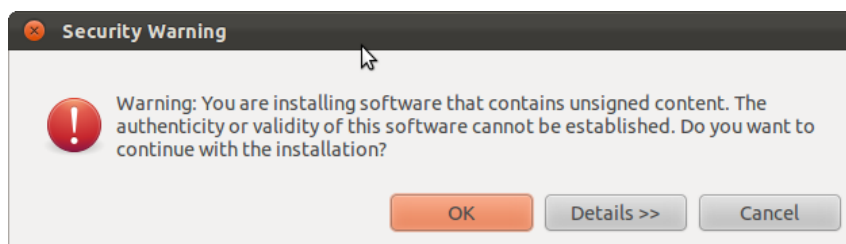
Acione o menu: Help -> Eclipse Marketplace, como mostra a figura abaixo:

Eclipse Marketplace

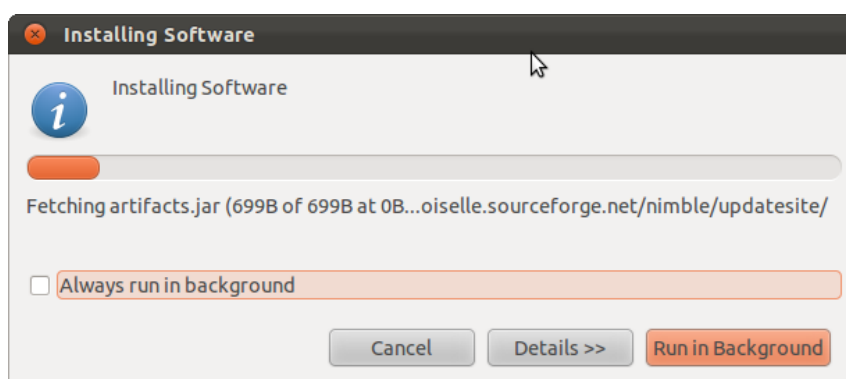
No próximo passo, digete a palavra Demoiselle no campo de busca e clique no botão *GO*

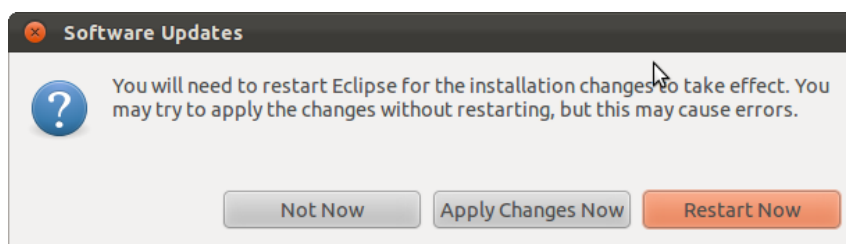
Após carregado o resultado da pesquisa localize o item *Demoiselle Nimble Eclipse Plugin* e clique no botão *Install*

Localizando Demoiselle Nimble no Eclipse Marketplace

Na tela seguinte, verifique se os itens estão selecionados e clique no botão *Next >*

Selecionando o plugin

Na tela seguinte, é preciso aceitar a licença, clicar no botão *Finish*

Aceitando a licença

Durante este processo, a tela abaixo pode ser apresentada, é apenas um aviso sobre a assinatura da bibliteca , basta clicar no botão *OK*

Aviso sobre assinatura da biblioteca

Depois aguarde o processo de instalação terminar

Aceitando a licença

Ao final o Eclipse precisará ser reiniciado.

Aceitando a licença

Depois de reiniciado, veja *Seção 4.2, “Instruções de Uso”*

4.1.2. Instalação em versões antigas.

Os pré-requisitos para o uso da Plugin são os mesmos para o *Modo Desktop* além é claro da IDE Eclipse instalada

No Eclipse 3.6 (Helios) ou superior acione o menu **Help** → **Install New Software**

No Eclipse 3.5 (Helios) ou anterior acione o menu **Help** → **Software Updates**

A tela seguinte deverá ser apresentada

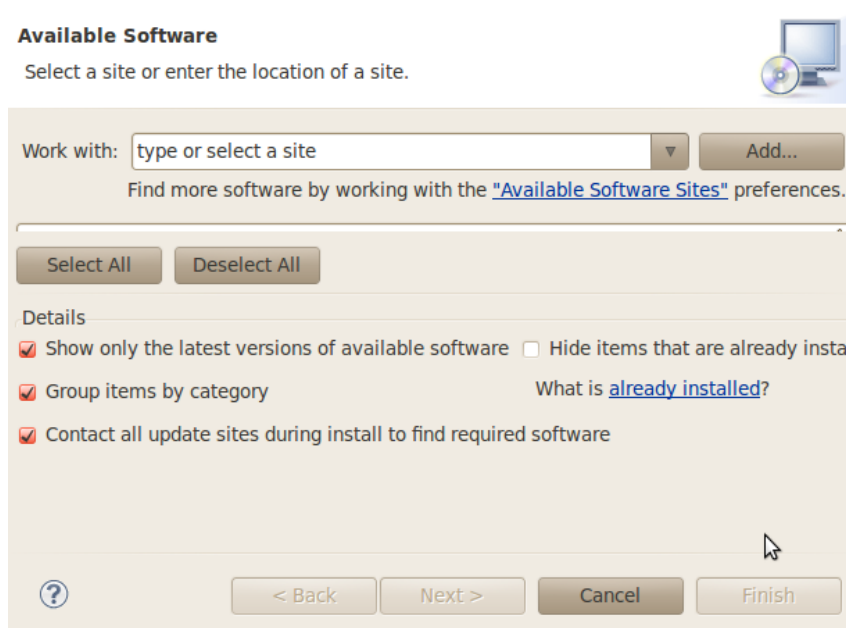
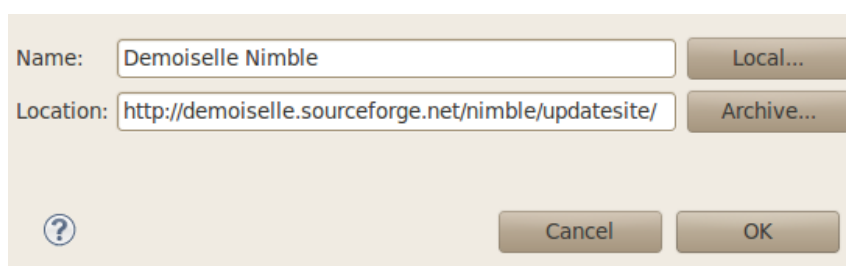



Imagem da Tela de instalação de Plugins para Eclipse

Clique no botão **Add...** para incluir um novo repositório, conforme a figura abaixo

Tela para inclusão de novo repositório de plugin

Nesta tela preecha as seguintes informações:

- **Name:**`Demoiselle Nimble`
- **Location:**`http://demoiselle.sourceforge.net/nimble/updatesite`

Voltando à tela anterior, que deverá ser atualizada assim:

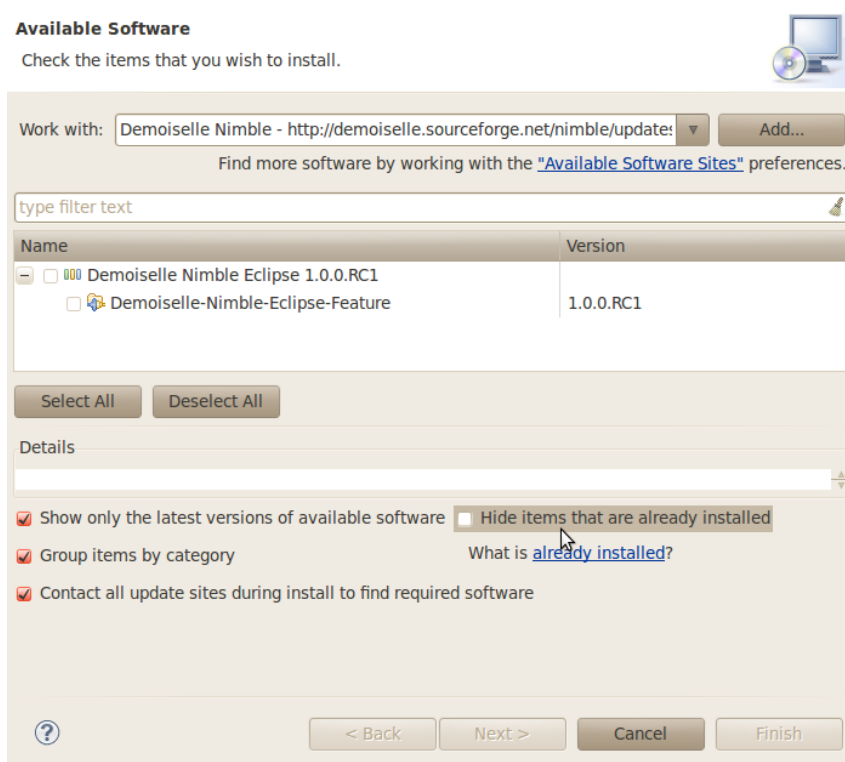



Imagem da Tela de instalação de Plugins para Eclipse, com novo repositório

No quadro onde aparecem as opções *Name* e *Version* Procure e marque a versão mais atual.Em seguida clique no botão **Next >**

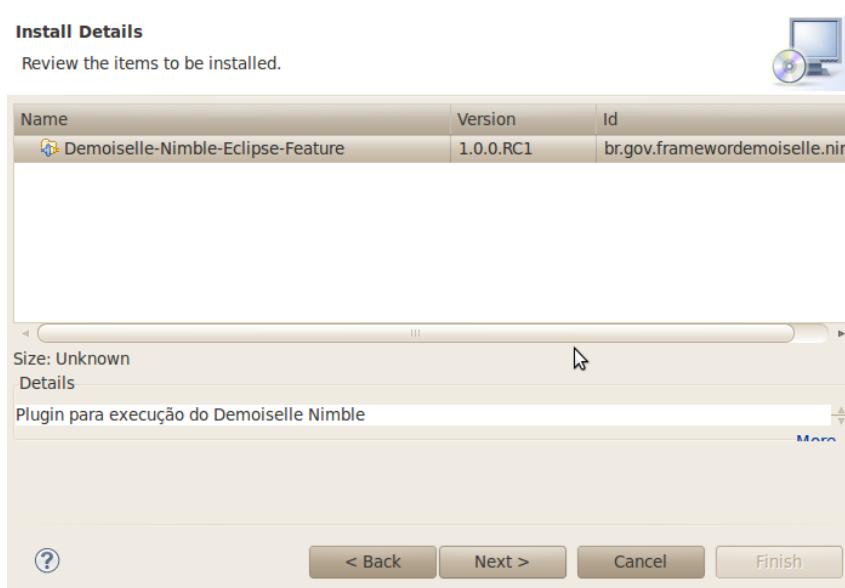



Imagem da Tela de detalhes da instalação do Plugin

Se não houver erros ou avisos, clique no botão **Next >**

Na última tela marque a opção de aceite da licença e clique no botão **Finish**

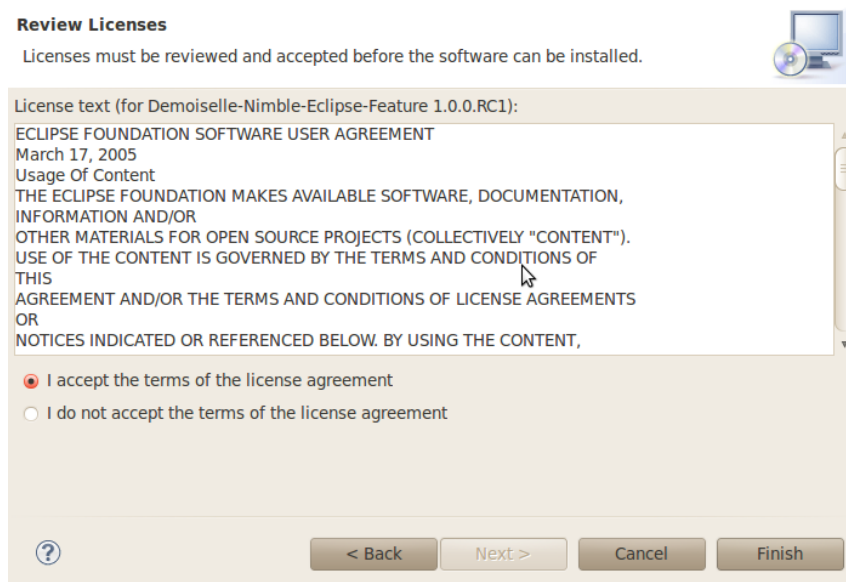



Imagem da Tela de aceite da licença

Aguarde a instalação e reinicie o Eclipse, conforme será sugerido pelo próprio

4.2. Instruções de Uso

Feita a instalação conforme as opções anteriores, já é possivel executar o Demoiselle Nimble, os acionadores estão em:

- Um ícone na barra de ferramentas:

Icone do Demoiselle Nimble na barra de ferramentas

- Em um Menu Próprio

Icone do Demoiselle Nimble na barra de ferramentas

Para saber mais sobre o uso com eclipse veja *Capítulo 6, Exemplo JSF/PrimeFaces usando Eclipse IDE*

Plugin para NetBeans IDE

Guia de uso para o *Puglin para NetBeansIDE*

Assim como no Eclipse o Demoiselle Nimble também tem a sua integração à *IDE NetBeans* [http://netbeans.org/downloads/] através de um plugin.

Como o Demoiselle-Nimble foi concebido para ser independente de Plataforma, o Netbeans faz apenas uma a chamada à interface gráfica do Demoiselle-Nimble.

Assim, as instruções de uso dos *templates* são as mesmas contidas no *Guia - Interface Gráfica*

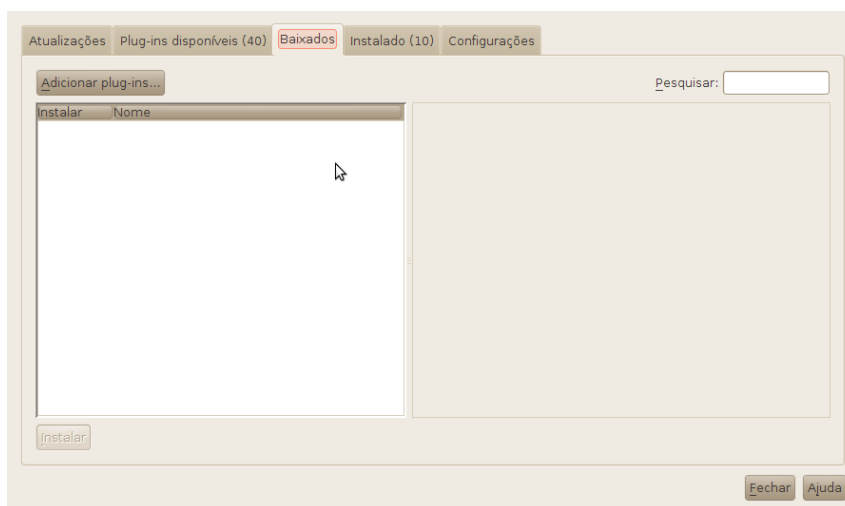
Neste capítulo vamos nos ater apenas os detalhes referentes à intalação e uso no NetBeans

5.1. Instalação

Os pré-requisitos para o uso da Plugin são os mesmos para o *Modo Desktop* além é claro da IDE NetBeans instalada (Testada com v 7.0)

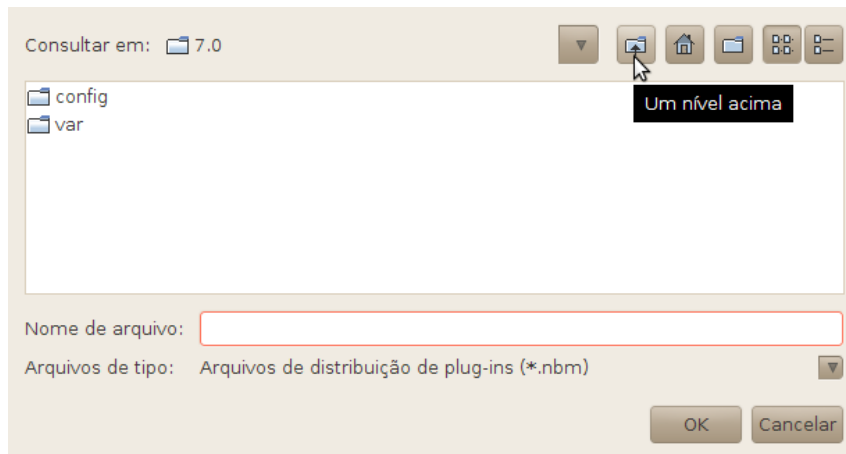
Baixe o arquivo de instalação do pluguin do endereço: *http://demoiselle.sourceforge.net/nimble/netbeans/br-gov-frameworkdemoiselle-tools-nimble-netbeans.nbm*

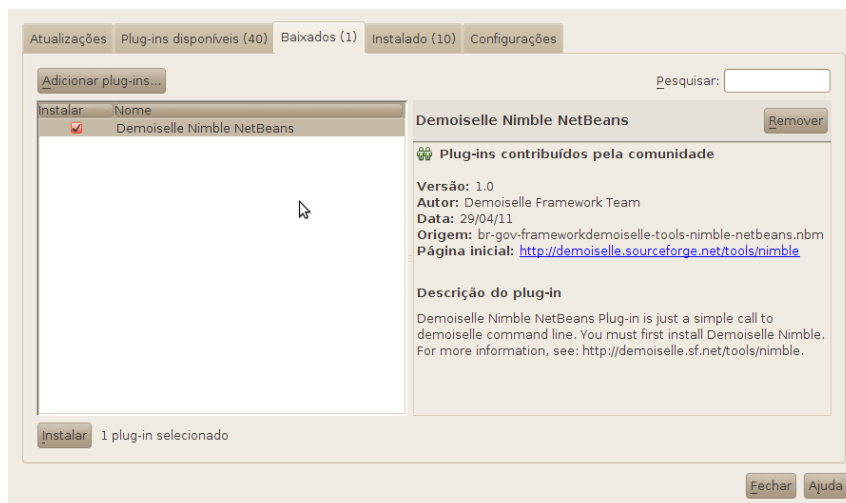
No ambiente de desenvolvimento do NetBeans acione o menu **Ferramentas** → **Plugins**

A tela seguinte deverá ser apresentada

Tela de instalação de Plugins para NetBeans

Selecione a aba **Baixados** e clique no botão

Adicionar Plug-ins...
para incluir um novo repositório, conforme a figura abaixo

Na tela abaixo selecione o arquivo*br-gov-frameworkdemoiselle-tools-nimble-netbeans.nbm* no diretório onde foi baixado o Plugin

Tela para inclusão de novo repositório de plugin

Voltando à tela anterior, que deverá ser atualizada assim:

Tela de instalação de Plugins para NetBeans, com novo plugin

No quadro onde aparecem as colunas **Instalar** e **Nome** Marque a coluna Instalar.Em seguida clique no botão com o mesmo nome

Instalar
no canto inferior esquerdo da tela

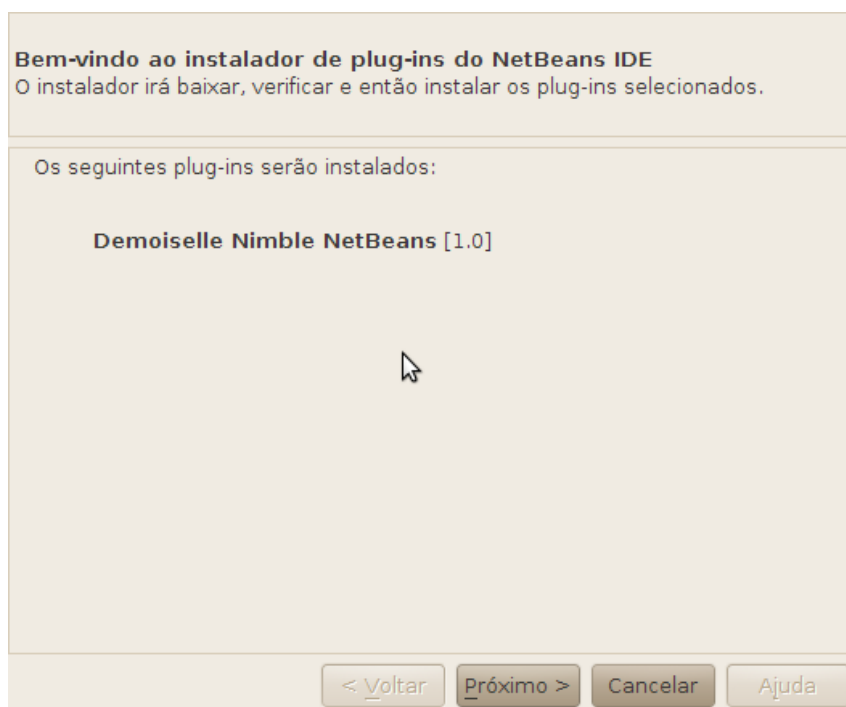



Imagem da Tela de detalhes da instalação do Plugin

Se não houver erros ou avisos, clique no botão

Próximo >

Na próxima tela marque a opção de aceite da licença e clique no botão

Instalar

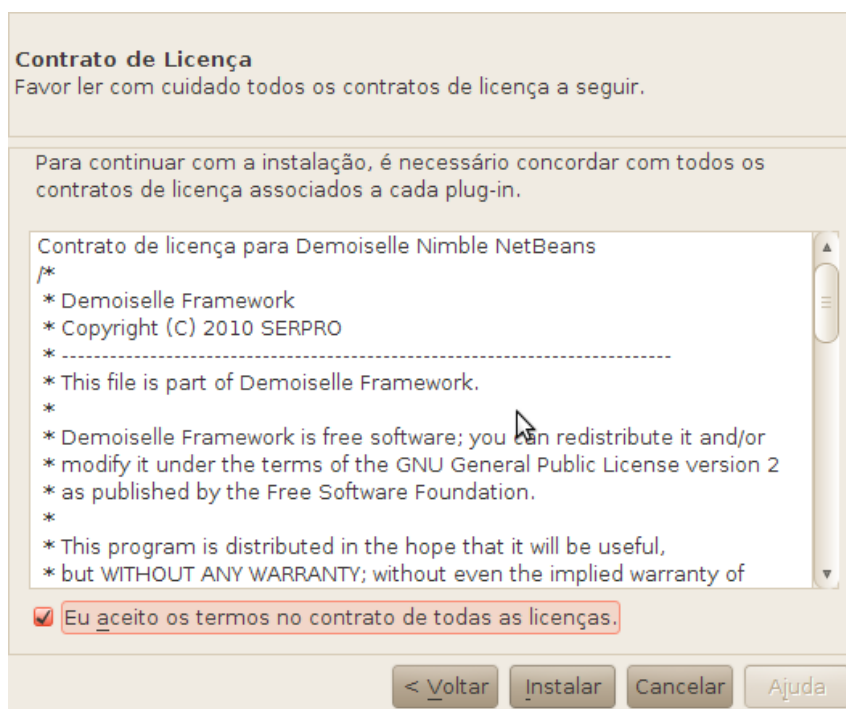
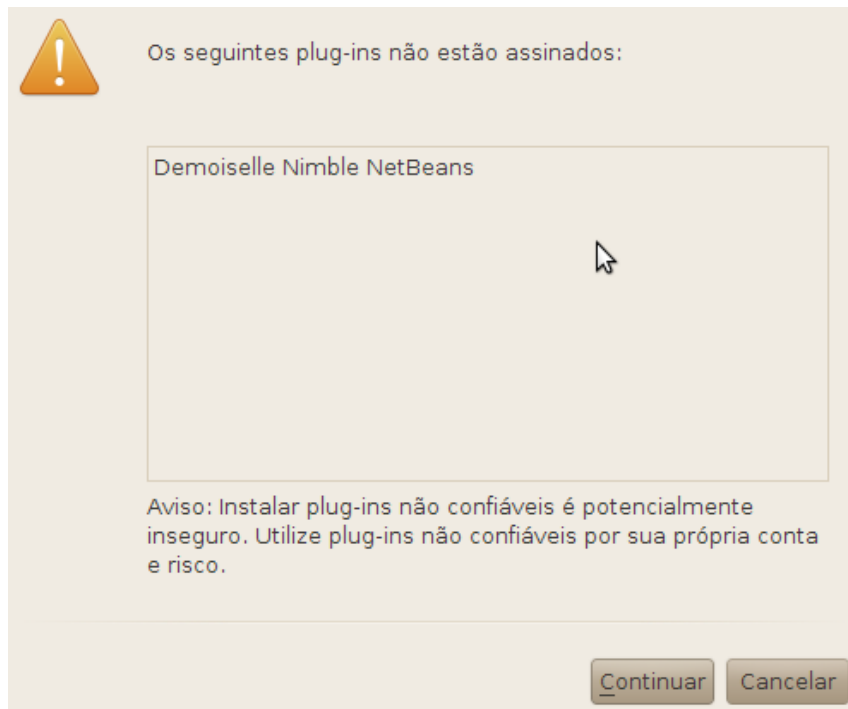



Imagem da Tela de aceite da licença

Na tela seguinte clique no botão

Continuar

Tela de sequência de instalação

Aguarde a instalação

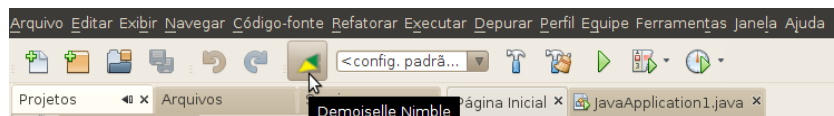
5.2. Instruções de Uso

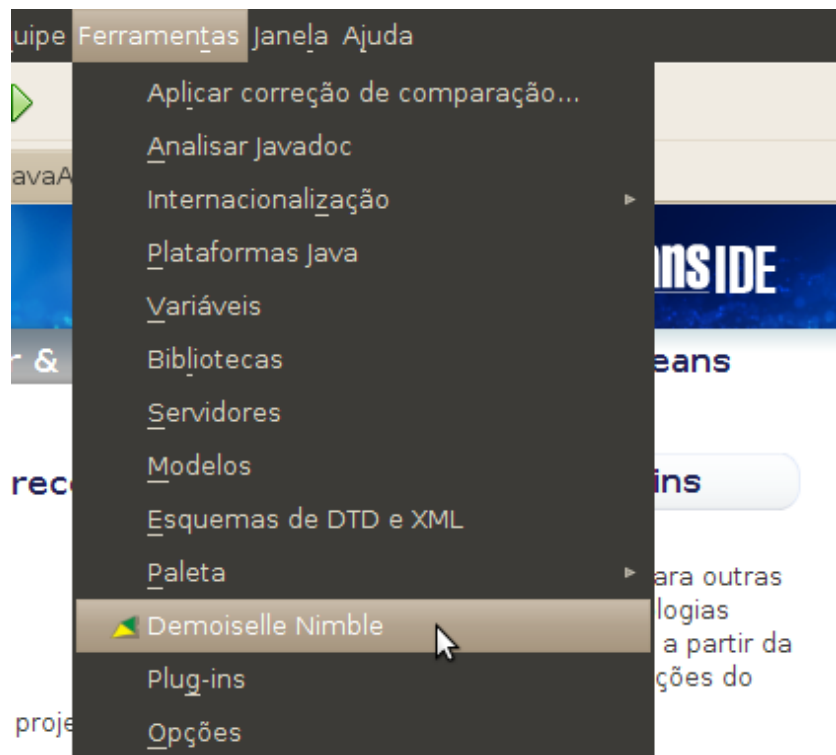
Feita a instalação conforme as opções anteriores, já é possivel executar o Demoiselle Nimble

Nesta versão do plugin é preciso que seja criado um projeto para que o plugin seja habilitado

Recomendamos usar o arquétipo Maven do Demoiselle *http://demoiselle.sourceforge.net/repository/archetype-catalog.xml*

Depois de criado, selecione o projeto e veja onde estão os acionadores do Plugin Demoiselle Nimble

- Um ícone na barra de ferramentas:

Icone do Demoiselle Nimble na barra de ferramentas

- Sub-item no Menu Ferramentas

Sub-item Demoiselle Nimble

Exemplo JSF/PrimeFaces usando Eclipse IDE

Exemplo prático usando o ambiente de Desenvolvimento *Eclipse IDE* para geração de aplicação JSF com Primefaces

O Demoiselle Nimble possui um plugin para Eclipse que já conta com uma boa integração. Então neste capítulo faremos uma desmonstração com o uso desta ferramenta.

Vamos considerar que a instalação e configuração já esteja de acordo com as intruções contidas na *Seção 4.1, "Instalação"*

6.1. Prepararação do Ambiente

Antes de iniciar o uso do Demoiselle-Nimble, podemos criar um projeto usando um arquétipo do Demoiselle. Esse procedimento também pode ser feito diretamente no Eclipse ou ou Maven sem o uso do Nimble, mas a opção do Demoiselle-Nimble serve para facilitar.

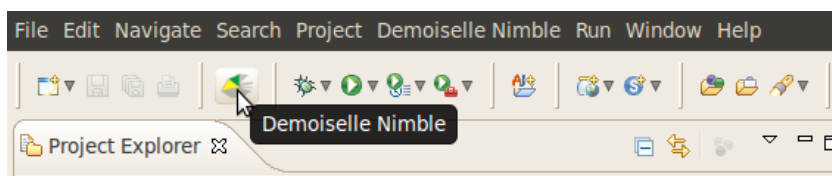
Abaixo video de exemplo de criação de projeto com Plugin Maven do Eclipse.

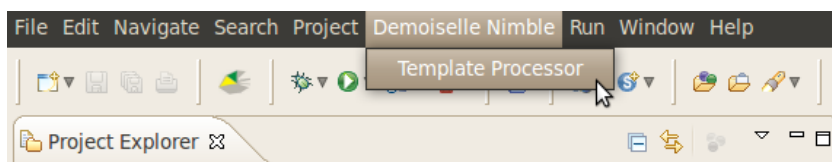
http://youtu.be/2jxzHqI1RCk

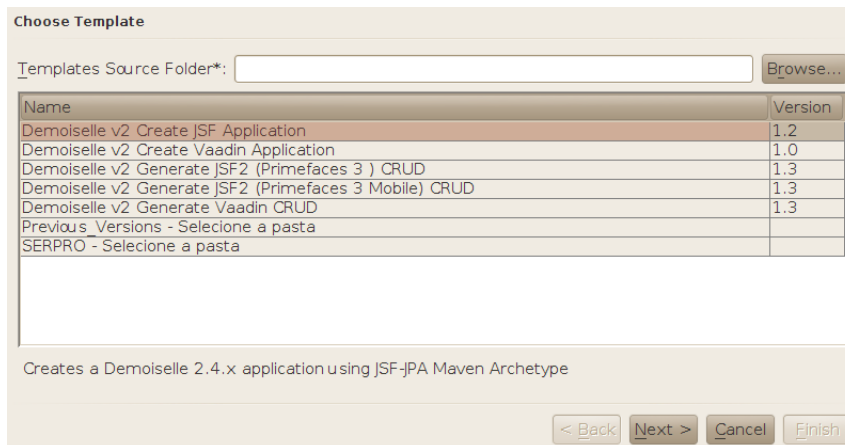
6.1.1. Criação do projeto base

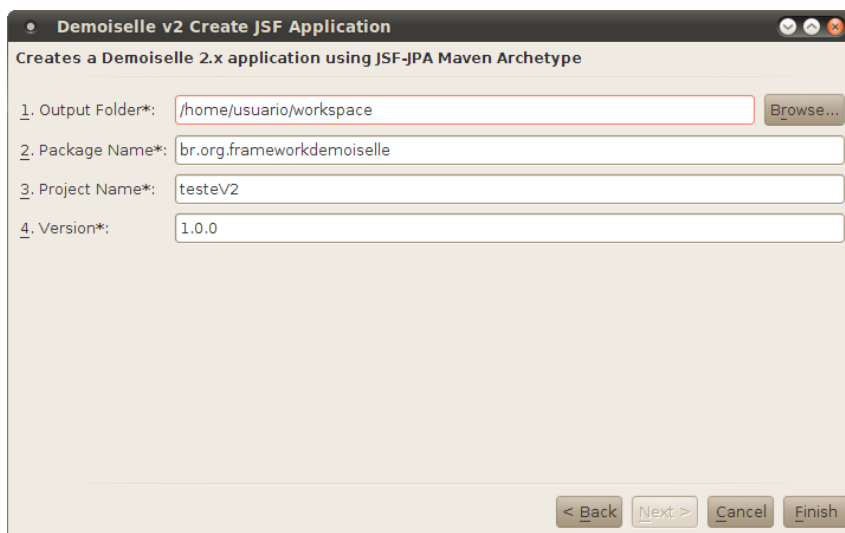
Caso opte por criar o projeto Maven através do próprio Eclipse vá direto para : *Seção 6.1.2, "Configurações e Classes de Domínio"*

Com o Eclise aberto, veja onde estão os acionadores do Plugin Demoiselle Nimble:

- Um ícone na barra de ferramentas:

Icone do Demoiselle Nimble na barra de ferramentas

- Sub-item no Menu Ferramentas

Sub-item Demoiselle Nimble

Acione o Demoiselle-Nimble através dos atalhos acima, após alguns segundos a tela seguinte deverá ser apresentada:

Criando uma aplicação Maven com Demoiselle Nimble

Selecione a opção: Demoiselle V2 Create JSF Application, e clique no botão *Next >*

Criando uma aplicação JSF com Demoiselle Nimble

Nesta tela, na opção: 1. Output Folder, no Eclipse estará selecionado o Workspace atual, caso contrário utilize o botão *Browse...* para encontrar o diretório correto

Informe as seguintes informações nos campos seguintes:

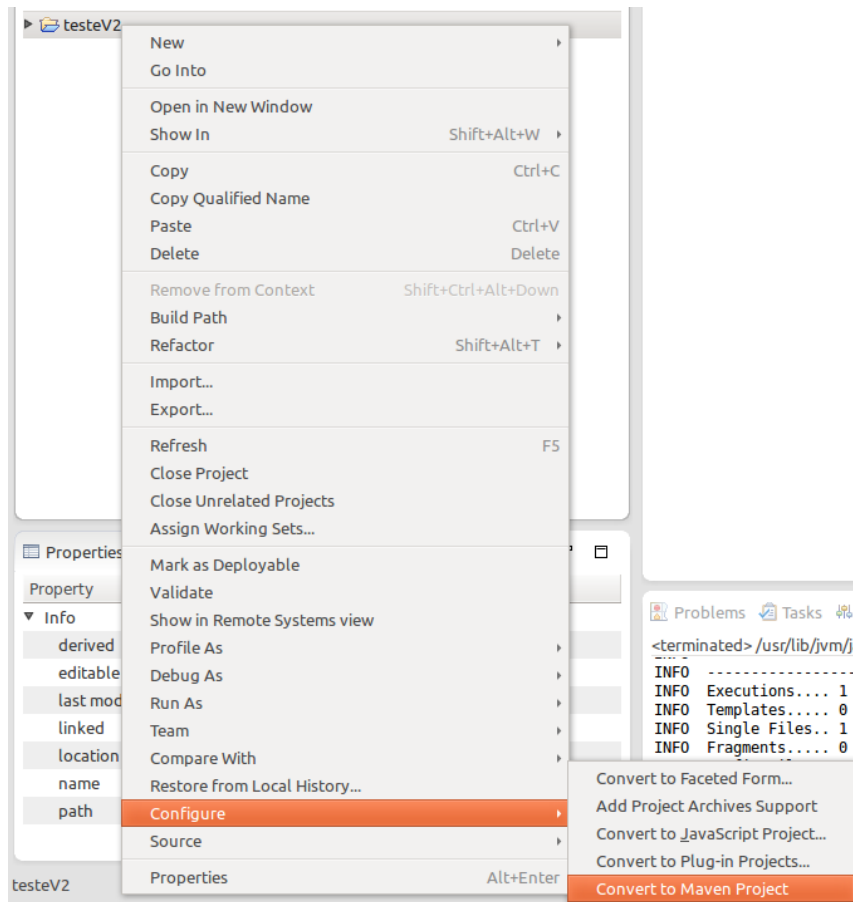
2. Package Name: br.org.frameworkdemoiselle

3. Project Name: testeV2

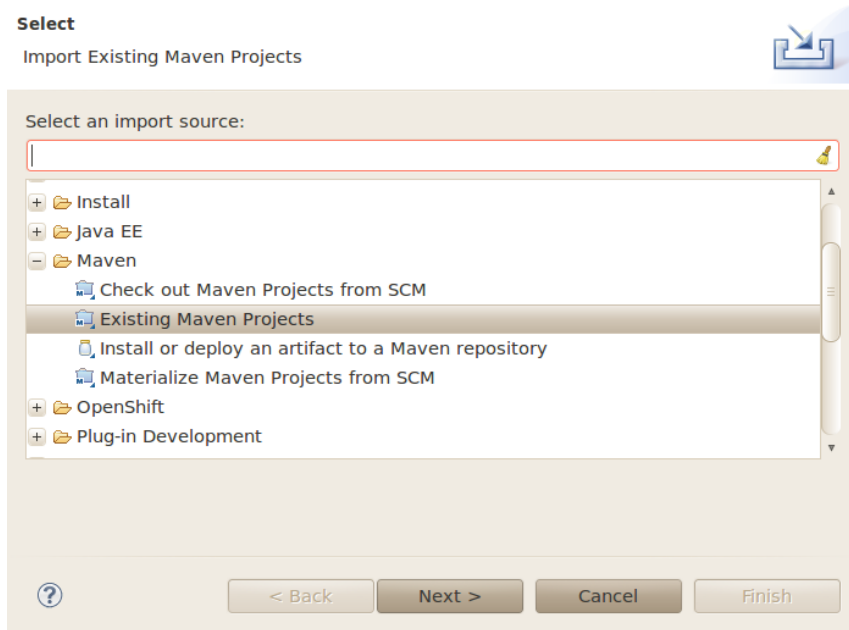
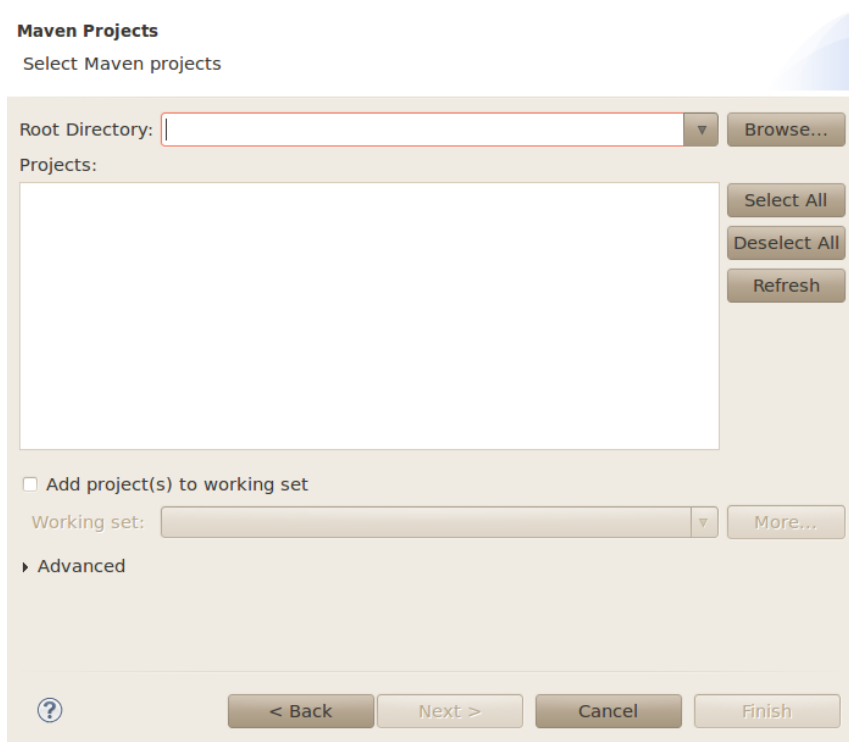
4. Version: 1.0.0

Em seguinda clique no botão *Finish* e aguarde que o processo seja concluído.

O projeto deverá aparecer na área de trabalho do Eclipse, mas como foi criado através do Nimble não está configurado como um projeto Maven. Para que o Eclipse reconheça como projeto Maven é preciso configurá-lo.

Para isso, selecione o projeto e clique com o botão direito do mouse para ativar o menu e selecionar as opções: Configure -> Convert to Manven Project

Configurando projeto para Maven

Com isso já podemos partir para *Seção 6.1.2, "Configurações e Classes de Domínio"*, mas podem haver caos (versões mais antigas do Eclipse) em que devemos importar o projeto Maven que foi criado para dentro do wokspace de trabalho. Se este for o caso veja os passos abaixo

Para importar o projeto, devemos usar a opção File -> Import e em seguida selecionar: Maven -> Existing Maven Project

Importando o projeto criado pelo Nimble para o Eclipse

Selecionando o diretório do projeto

Na tela acima, selecione o diretório onde o projeto foi criado e depois clique no botão finish e aguarde o Eclipse fazer a importação do projeto.

Abaixo o video de demonstração de criação de projeto com apoio do Demoiselle Nimble.

http://youtu.be/witf3ZIhLIM

6.1.2. Configurações e Classes de Domínio

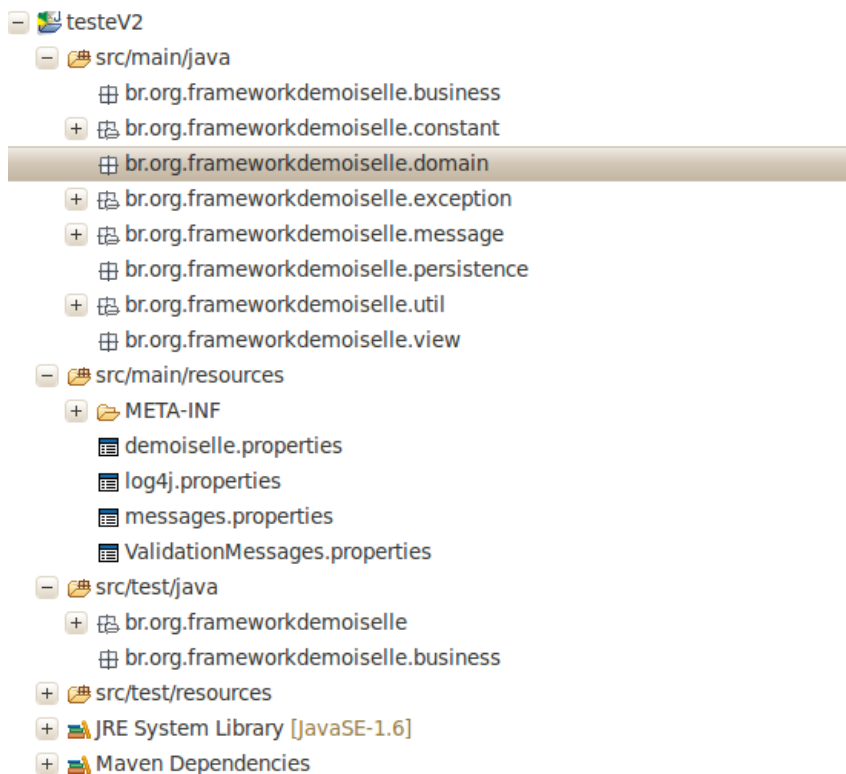
Com um projeto Demoiselle JSF-JPA criado, vamos verificar algumas configurações e fazer a criação das classes de domínio necessárias para o uso do Demoiselle-Nimble

Abra o projeto e verifique o arquivo /src/main/resources/persistence.xml e remova os comentários para escolher a estratégia de transação. Conforme exemplo mostrado abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/
persistence/persistence_2_0.xsd">

    <!-- If you are using JBoss AS7 with non JTA transaction then use this persistence-unit -->
    <!--
    -->
    <persistence-unit name="testeV2-ds" transaction-type="RESOURCE_LOCAL">
        <non-jta-data-source>java:jboss/datasources/ExampleDS</non-jta-data-source>
        <properties>
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="false" />
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
        </properties>
    </persistence-unit>
</persistence>
```

No arquivo /src/main/resources/messages.properties, encontre e mude a propriedade main.app.title para o titulo da sua aplicação, como no exemplo abaixo:

main.app.title=TesteV2

Em seguida, vamos localizar o pacote *Domain* conforme a figura abaixo:

Localizando o pacote das classes de domínio

Crie neste pacote uma classe chamada *Pessoa*, que será uma classe abstrata apenas para que possamos testar com o conceito de herança, conforme o código abaixo

```
import java.util.Date;
import javax.persistence.*;

@MappedSuperclass
public abstract class Pessoa {

    @Id
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @Column
    private Long cpf;

    @Column(nullable=false,  length=255)
    private String nome;


    @Column(nullable=false)
    @Temporal(value=TemporalType.DATE)
    private Date dataNascimento;

    @Enumerated(EnumType.STRING)
    @Column (length=10)
    private Genero genero;

    public Pessoa(){
```

```
        super();
    }

    public Pessoa(Long cpf, String nome, Date dataNascimento, Genero genero) {
        this.cpf = cpf;
        this.nome = nome;
        this.dataNascimento = dataNascimento;
        this.genero = genero;
    }

    public void setId(Long id) {
        this.id = id;
    }
    public Long getId() {
        return id;
    }
    public void setNome(String nome) {
        this.nome = nome;
    }
    public String getNome() {
        return nome;
    }
    public void setCpf(Long cpf) {
        this.cpf = cpf;
    }
    public Long getCpf() {
        return cpf;
    }
    public void setDataNascimento(Date dataNascimento) {
        this.dataNascimento = dataNascimento;
    }
    public Date getDataNascimento() {
        return dataNascimento;
    }
    public Genero getGenero() {
        return genero;
    }
    public void setGenero(Genero genero) {
        this.genero = genero;
    }
}
```

Depois a classe *Estudante*, que extenderá Pessoa de acordo com o código abaixo :

```
import java.io.Serializable;
import java.util.Date;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name="TB_Estudante")
public class Estudante extends Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;

    @Column(nullable=false)
```

```
    @Temporal(value=TemporalType.DATE)
    private Date dataMatricula;

    @Column
    private Integer numeroMatricula;

    @ManyToOne
    @JoinColumn (name= "turma_fk")
    private Turma turma;

    @OneToOne (cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    private BolsaEstudo bolsaEstudo;


    @ManyToMany (cascade = CascadeType.ALL, fetch = FetchType.LAZY)
   @JoinTable(name = "tb_estudante_endereco", joinColumns = @JoinColumn(name = "id_estudante"),
               inverseJoinColumns = @JoinColumn(name = "cod_endereco"))
    private List<Endereco> enderecos;

    public Estudante(){
        super();
    }

    public Estudante(Date dataMatricula, Integer numeroMatricula, Long cpf, String nome,
                Date dataNascimento, Genero genero) {
        super(cpf, nome, dataNascimento, genero);
        this.dataMatricula = dataMatricula;
        this.numeroMatricula = numeroMatricula;
    }


    public Estudante(Date dataMatricula, Integer numeroMatricula, Turma turma,
                     BolsaEstudo bolsaEstudo, List<Endereco> enderecos, Long cpf,
                     String nome, Date dataNascimento, Genero genero) {
        super(cpf, nome, dataNascimento, genero);
        this.dataMatricula = dataMatricula;
        this.setTurma(turma);
        this.numeroMatricula = numeroMatricula;
        this.bolsaEstudo = bolsaEstudo;
        this.setEnderecos(enderecos);

    }

    public Date getDataMatricula() {
        return dataMatricula;
    }

    public void setDataMatricula(Date dataMatricula) {
        this.dataMatricula = dataMatricula;
    }

    public Integer getNumeroMatricula() {
        return numeroMatricula;
    }

    public void setNumeroMatricula(Integer numeroMatricula) {
        this.numeroMatricula = numeroMatricula;
    }
```

```
    public Turma getTurma() {
        return turma;
    }

    public void setTurma(Turma turma) {
        this.turma = turma;
    }

    public BolsaEstudo getBolsaEstudo() {
        return bolsaEstudo;
    }

    public void setBolsaEstudo(BolsaEstudo bolsaEstudo) {
        this.bolsaEstudo = bolsaEstudo;
    }

    public List<Endereco> getEnderecos() {
        return enderecos;
    }

    public void setEnderecos(List<Endereco> enderecos) {
        this.enderecos = enderecos;
    }

}
```

Teremos a classe de enumeração Genero usada por "Pessoa":

```
public enum Genero {

    MALE, FEMALE;
}
```

E as demais classes necessárias para o exemplo:

- BolsaEstudo.java:

```
import java.io.Serializable;
import java.util.Date;
import javax.persistence.*;

@Entity
@Table(name = "TB_Bolsa")
public class BolsaEstudo implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "id_bolsa")
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Integer numero;

    @Column
    private Long porcentagem;
```

```
    @Column(nullable = false)
    @Temporal(value = TemporalType.DATE)
    private Date dataInicio;

    @Column
    @Temporal(value = TemporalType.DATE)
    private Date dataFim;

    public BolsaEstudo() {
        super();
    }

    public BolsaEstudo(Long porcentagem, Date dataInicio,
            Date dataFim) {
        super();
        this.porcentagem = porcentagem;
        this.dataInicio = dataInicio;
        this.dataFim = dataFim;
    }

    public Integer getNumero() {
        return numero;
    }

    public void setNumero(Integer numero) {
        this.numero = numero;
    }

    public Long getPorcentagem() {
        return porcentagem;
    }

    public void setPorcentagem(Long porcentagem) {
        this.porcentagem = porcentagem;
    }

    public Date getDataInicio() {
        return dataInicio;
    }

    public void setDataInicio(Date dataInicio) {
        this.dataInicio = dataInicio;
    }

    public Date getDataFim() {
        return dataFim;
    }

    public void setDataFim(Date dataFim) {
        this.dataFim = dataFim;
    }

}
```

- Endereco.java:

```
import java.io.Serializable;
import java.util.List;
import javax.persistence.*;

@Entity
@Table(name = "TB_Endereco")
public class Endereco implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "cod_endereco")
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long codigo;

    @Column
    private String logradouro;

    @Column
    private String cidade;

    @Column
    private String estado;

    @Column
    private String cep;

    @ManyToMany(mappedBy="enderecos")
    private List<Estudante> estudantes;

    public Endereco (){
        super();
    }
    public Endereco(String logradouro, String cidade,
            String estado, String cep) {
        super();
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.cep = cep;
    }
    public Endereco(String logradouro, String cidade,
            String estado, String cep, List<Estudante> estudantes) {
        super();
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.cep = cep;
        this.estudantes = estudantes;
    }
    public Long getCodigo() {
        return codigo;
    }
    public void setCodigo(Long codigo) {
        this.codigo = codigo;
    }
    public String getLogradouro() {
        return logradouro;
```

```
    }
    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }
    public String getCidade() {
        return cidade;
    }
    public void setCidade(String cidade) {
        this.cidade = cidade;
    }
    public String getEstado() {
        return estado;
    }
    public void setEstado(String estado) {
        this.estado = estado;
    }
    public String getCep() {
        return cep;
    }
    public void setCep(String cep) {
        this.cep = cep;
    }   public List<Estudante> getEstudantes() {
        return estudantes;
    }
    public void setEstudantes(List<Estudante> estudantes) {
        this.estudantes = estudantes;
    }
}
```

- Turma.java:

```
import java.io.Serializable;
import java.util.*;
import javax.persistence.*;
import javax.validation.constraints.NotNull;

@Entity
@Table(name = "TB_Turma")
public class Turma implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @Column(name = "id_turma")
    @GeneratedValue(strategy = GenerationType.SEQUENCE)
    private Long id;

    @Column(length = 255)
    @NotNull
    private String nomeTurma;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY)
    @JoinColumn(name = "turma_fk")
    private List<Estudante> estudantes = new ArrayList<Estudante>();

    public Turma() {
```

```
        super();
    }

    public Turma(String nomeTurma) {
        this.nomeTurma = nomeTurma;
    }

    public Turma(String nomeTurma, List<Estudante> estudantes) {
        this.nomeTurma = nomeTurma;
        this.estudantes = estudantes;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getNomeTurma() {
        return nomeTurma;
    }
    public void setNomeTurma(String nomeTurma) {
        this.nomeTurma = nomeTurma;
    }
    public List<Estudante> getEstudantes() {
        return estudantes;
    }
    public void setEstudantes(List<Estudante> estudantes) {
        this.estudantes = estudantes;
    }
}
```

Isso é tudo que precisamos criar manualmente para começar o uso do Demoiselle-Nimble. Caso tenha uma base de dados pronta, é possível utilizar um utilitário como o *Hibernate Tools* [http://hibernate.org/tools/] para fazer a geração das classes mapeadas com JPA.

A única exigência para geração de código com o Demoiselle-Nimble, é que as classes estejam com as anotações de JPA, conforme o exemplo acima. E a ferramenta atuará sobre as classes anotadas com @Entity.

Abaixo o video de demonstração da preparação do ambiente.

http://youtu.be/NdCP_7DIBjU

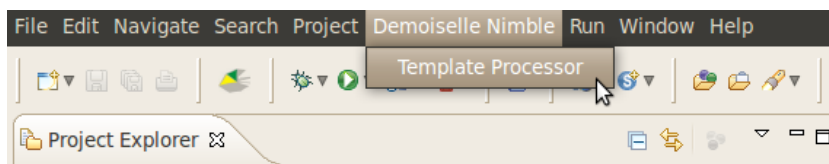
6.2. Gerando uma aplicação Web Tradicional

Neste primeiro exemplo, vamos utilizar a ferramenta para gerar os artefatos de uma aplicação WEB tradicional.

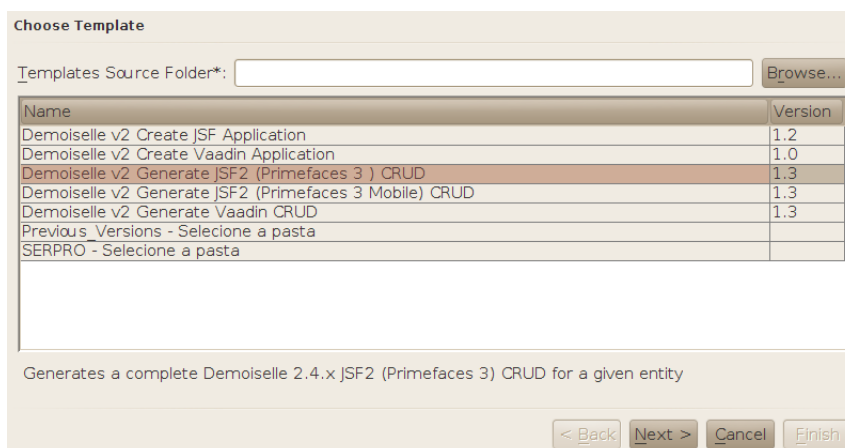
Selecione o projeto e veja onde estão os acionadores do Plugin Demoiselle Nimble

- Um ícone na barra de ferramentas:

Icone do Demoiselle Nimble na barra de ferramentas

- Sub-item no Menu Ferramentas

Sub-item Demoiselle Nimble

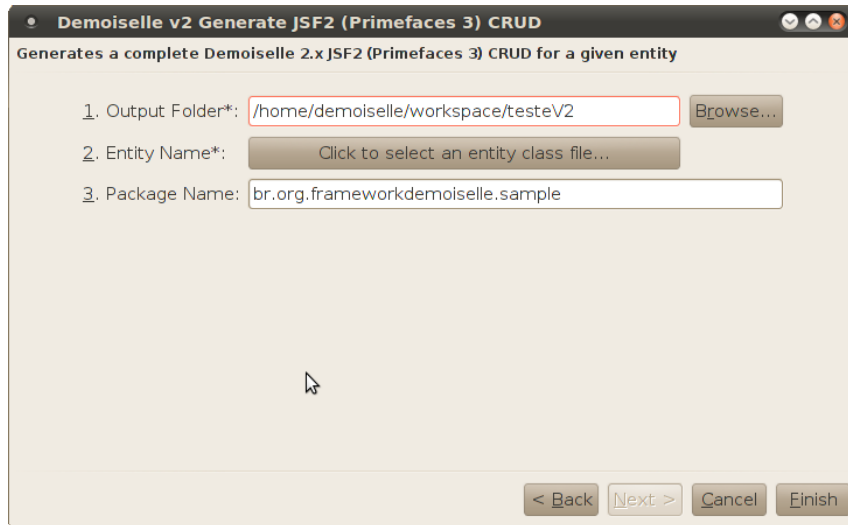
Com o projeto selecionado, use uma das opções acima para acionar a interface do Demoiselle Nimble, onde selecionaremos o Template para geração de um CRUD (Create, Read, Update e Delete) que são as operações básicas de criar, ler, atualizar e apagar

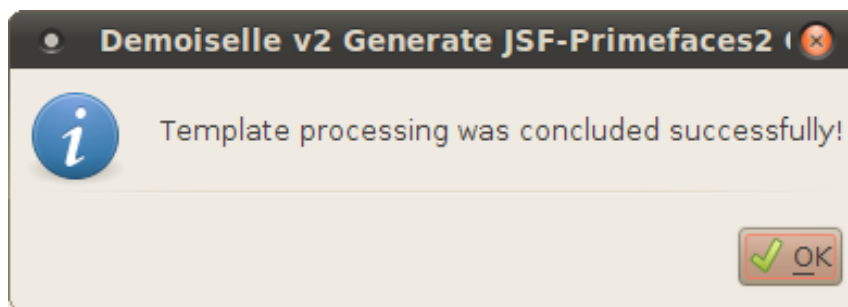
Selecione o Template adequado ao arquétipo que foi criado, neste exemplo optamos pelo Primefaces 3, que é compatível com a versão 2.4.0 (ou superior) do Demoiselle conforme a figura abaixo:

Escolhendo o template de CRUD

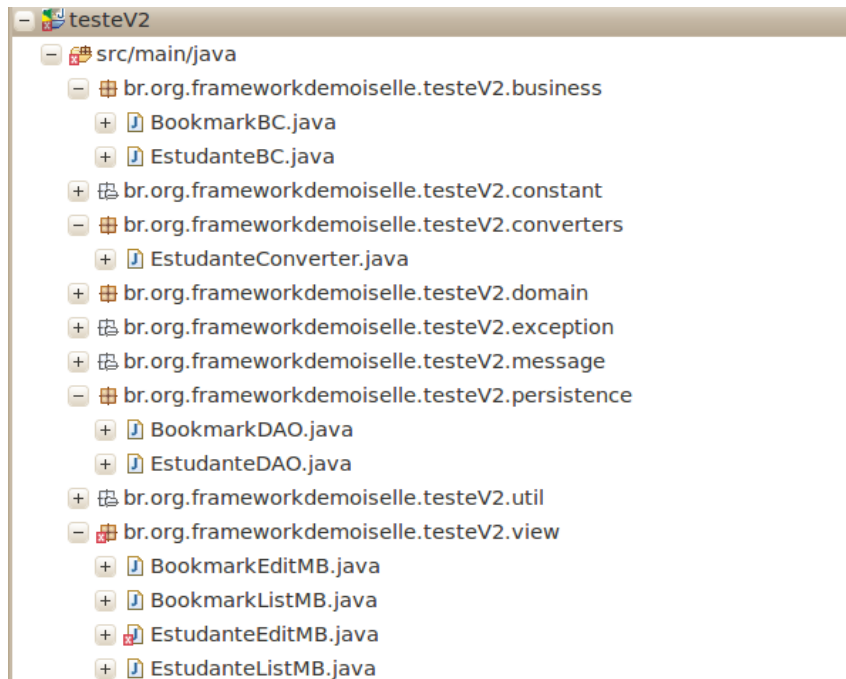
Na tela seguinte, deverão ser informados os parâmetros para geração dos artefatos

1. Output Folder*: Use o botão *Browse...* para selecionar o projeto dentro do diretório de Workspace do Eclipse

2. Entity Name*: Use o botão *Click to select an entity class file...* e procure no diretório (/src/main/java/br/org/frameworkdemoiselle/domain/) a classe Estudante.java

3. Package Name*: Será preenchido automáticamente com *br.org.frameworkdemoiselle*

Informando os parâmetros para criação dos artefatos

Aguarde o processamento até que a tela de confirmação apareceça

Processamento do Template OK

Confira os artefatos que foram gerados nos pacotes de negócio(business), persistência (persistence) e de visão (view) e no pacote de conversores (converters) que é específico para o JSF

Listagem dos artefatos criados em /src/main/java/

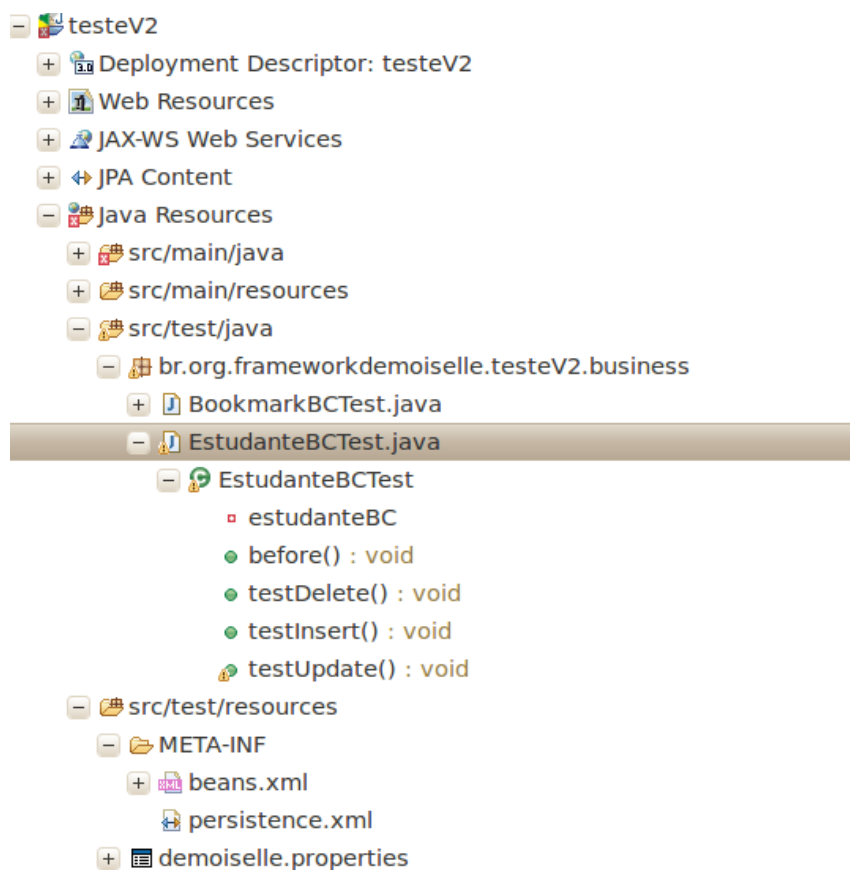
Na classe EstudanteEditMB é exibido um erro pois a classe Estudante possui relação com outras classes e ainda não criamos todos os artefatos para todas as classes de domínio, quando terminarmos não haverá mais esse erro.

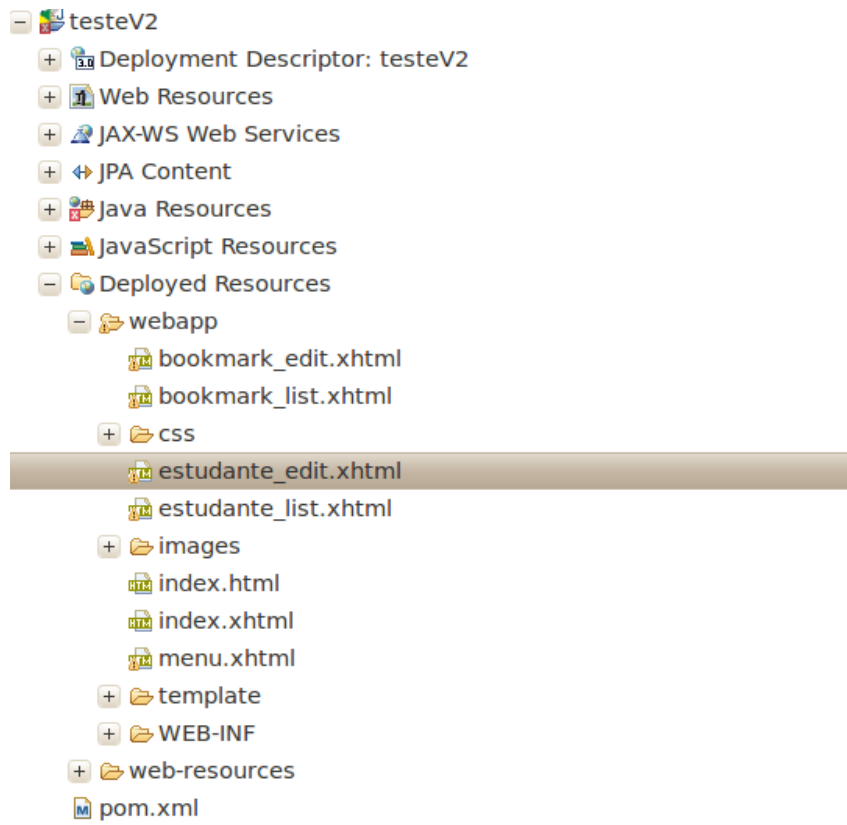
Veja também nos arquivos: persistence.xml e messages.properties, que novas informações foram incluídas.

Arquivo persistence.xml

Arquivo messages.properties

Na pasta /src/test/java/ será gerada a classe de teste de unidade para a classe de Negócio (BC) com testes para os métodos de inclusão (Insert), exclusão (Delete) e atualização (Update) de forma bastante simplificada.

O arquivo persistence.xml também será atualizado.

Listagem dos artefatos criados em /src/test/

Na pasta /src/main/webapp (Deployed Resources) também deverão ser geradas as páginas XHTML de edição e listagem e menu.xhml

Listagem dos artefatos criados em /src/main/webapp/

Repita os passos de geração para cada uma das outras classes de domínio que foram criadas na preparação do ambiente:

- BolsaEstudo.java
- Endereco.java
- Turma.java

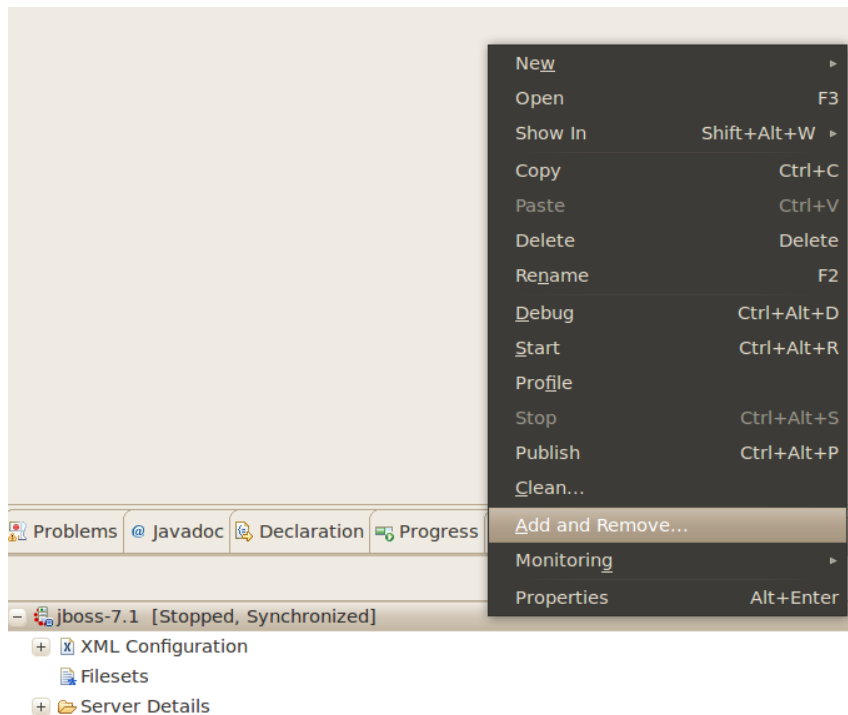
Abaixo o video de demonstração da criação dos artefatos CRUD.

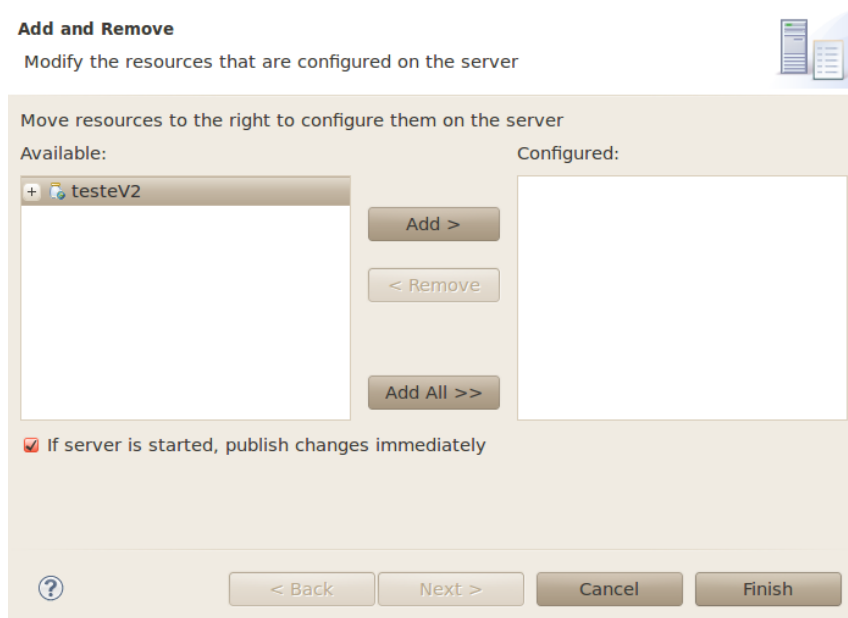
http://youtu.be/7YA5l7UsCzM

A partir da versão 1.2.0 do Demoiselle Nimble, é possível montar templates que identificam as anotações de relacionamentos do JPA, e assim podem ser definidos os componentes de tela os métodos de visão e negócio para tratar cada um deles. O template para JSF padrão trata os relacionamentos da segunte forma:

- OneToOne: ComboBox
- OneToMany: ComboBox
- ManyToOne: DataTable
- ManyToMany: PickList

Para testar as funcionalidade criadas, adicionaremos o projeto no servidor JBoss.

Localize a aba *"Servers"* e clique com o botão direito do mouse sobre o item: jboss-7.1 para acionar o menu. Em seguida selecione a opção *"Add and Remove..."*

Deploy da aplicação no servidor JBoss.

Na tela seguinte selecione o projeto clique primeiro no botão *Add* e depois *Finish*

Deploy da aplicação no servidor JBoss.

Com isso podemos iniciar o servidor de aplicação.

No exemplo a aplicação estará em http://localhost:8080/testeV2

Resultado na aplicação em funcionamento

Resultado na aplicação em funcionamento

Resultado na aplicação em funcionamento

Resultado na aplicação em funcionamento

Abaixo o video de demonstração da execução da aplicação criada pelo Demoiselle Nimble.

http://youtu.be/Dou9GbNtsNk

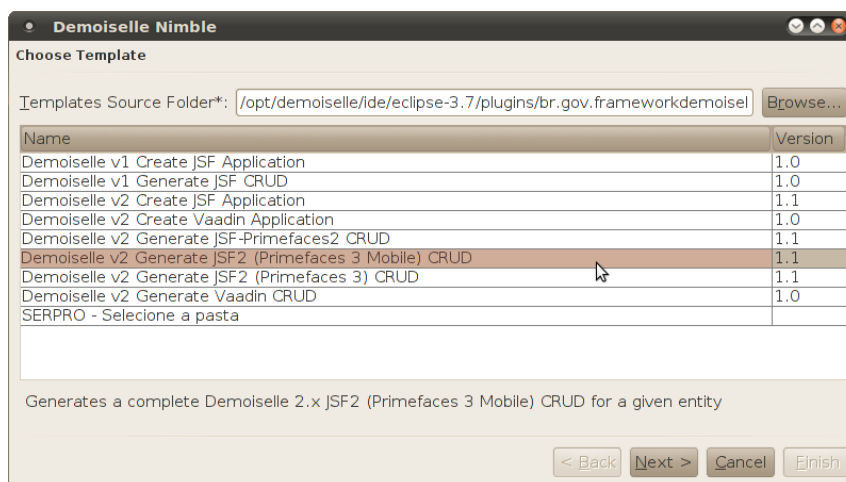
Exemplo de Web-Mobile (Primefaces) usando Eclipse IDE

Exemplo prático usando o ambiente de Desenvolvimento *Eclipse IDE* para geração de aplicação Web-Mobile com PrimeFaces

Vamos considerar que a instalação e configuração já esteja de acordo com as intruções contidas na *Seção 4.1, “Instalação”*. Também é preciso que seja feita a parte de preparação do ambiente conforme o inicio da *Seção 6.1, “Prepararação do Ambiente”*

7.1. Gerando uma aplicação Web Mobile

Atualmente muitos dos dispositivos de comunicação móveis (celulares, tablets, etc..) possuem acesso à internet, e é possível acessar através de um navegador (browser) qualquer aplicação ou site. Porém, na maioria dos casos a tela não fica bem apresentável pois não foi desenhada para um dispositivo móvel. Mas já existe soluções para isso, que são a bibliotecas para criar sites que se adaptam melhor neste tipo de dispositivo. Com base nisto o Demoiselle-Nimble possui um template (modelo) para geração deste tipo de interface.

Os procedimentos iniciais, que é iniciar a ferramenta, são os mesmos do item anterior: *Seção 6.2, “ Gerando uma aplicação Web Tradicional ”* mudando a partir da escolha do template. Conforme apresentado na tela abaixo:

Escolhendo o template de CRUD para Mobile

Na tela seguinte, deverão ser informados os parâmetros para geração dos artefatos

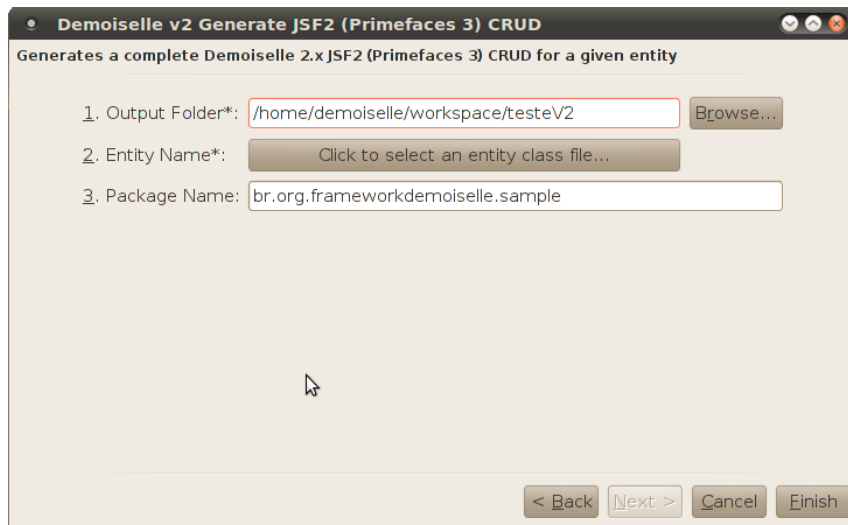
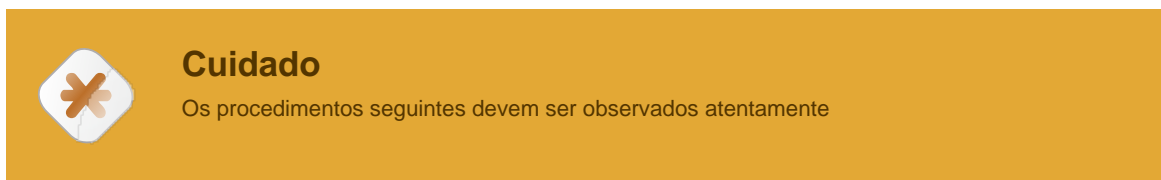
1. Output Folder*: Use o botão *Browse...* para selecionar o projeto dentro do diretório de Workspace do Eclipse

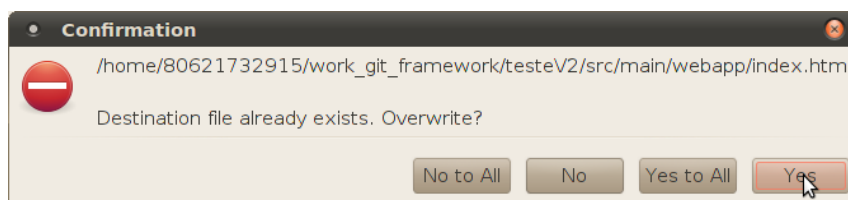
2. Entity Name*: Use o botão *Click to select an entity class file...* e procure no diretório (/src/main/java/br/org/frameworkdemoiselle/domain/) a classe Estudante

3. Package Name*: Será preenchido automáticamente com *br.org.frameworkdemoiselle*

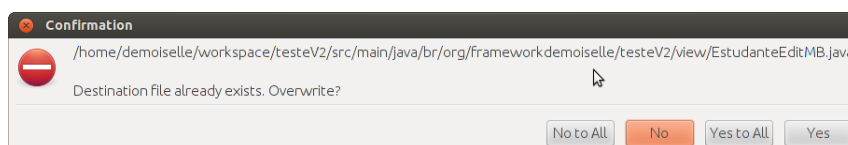
Informando os parâmetros para criação dos artefatos

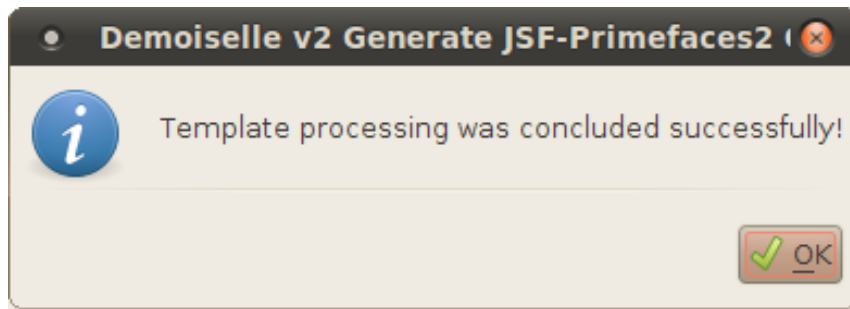
Cuidado

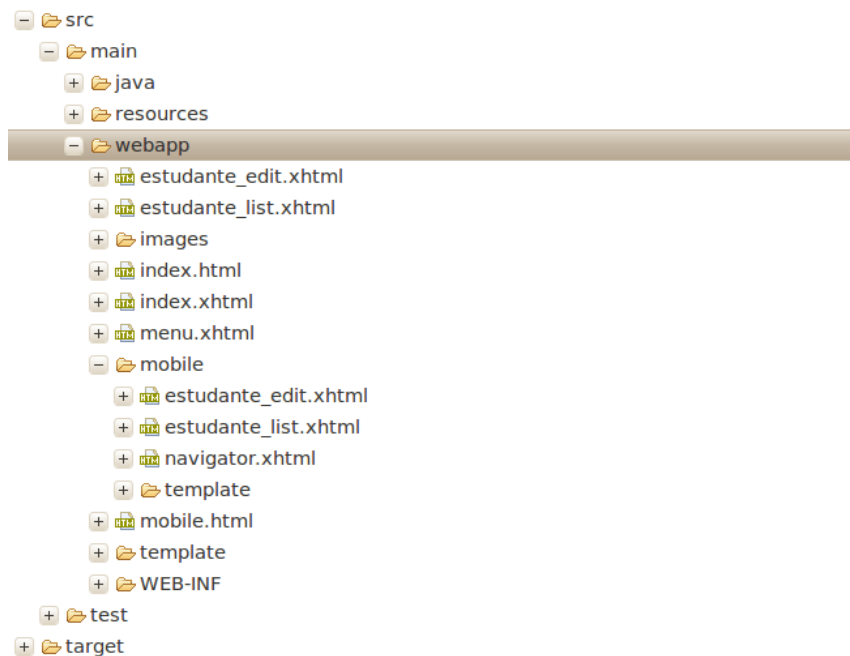
Os procedimentos seguintes devem ser observados atentamente

Ao clicar no botão *Finish* da tela anterior, algumas mensagens serão apresentadadas

A primeira será perguntando sobre o arquivo index.html, como estamos gerando uma aplicação Mobile o template poderá sobrepor esse arquivo para implementar uma forma bem simplificada de identificar qual tipo de dispositivo está acessando a aplicação. Neste caso é só uma forma de gerar um exemplo, mas caso sua aplicação possua uma forma mais elaborada, basta responder negativamente (clique no botão:*No*) ao pedido de sobrescrição mostrado na página abaixo, no caso deste exemplo vamos permitir que sobrescreva clicando no botão: *Yes*

Sobrescrevendo o arquivo index.html

Neste exemplo, a sequência mostrará uma tela que pergunta para sobrescrever as classes java de Persistência (DAO), Negócio (BC) e Visão(MB). Pois já havíamos criado essas classes no item anterior, caso esteja gerando apenas para mobile essa pergunta não aparecerá e as classes serão criadas. Note que são exatamente as mesmas pois a única diferença é justamente a parte de visão que são as páginas.

No nosso caso, vamos ignorar isso clicando no botão *No to All*

Sobrescrevendo as classes java

Ao final será apresentada a tela informando que o processamento foi executado

Processamento do Template OK

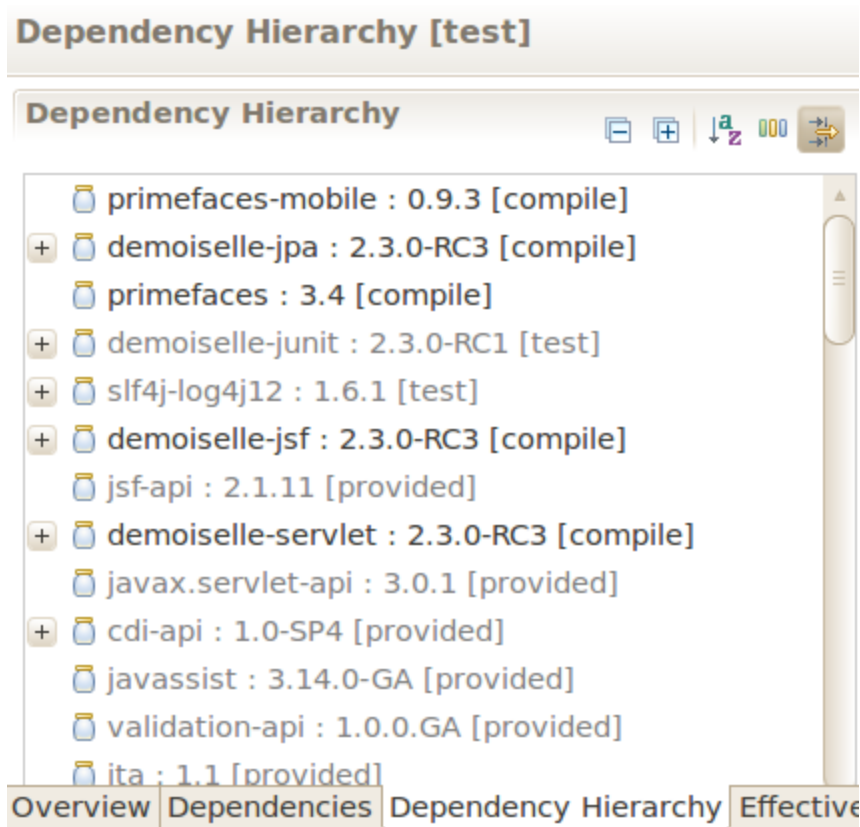
Com isso poderemos conferir no diretório /src/main/webapp/ os artefatos que foram criados:

Artefatos para web-mobile criados.

Além o novo arquivo index.html, também podemos verificar que foi criado o arquivo mobile.html. E um novo diretório (/mobile) também foi incluído e conterá todas páginas para a interface mobile.

Antes de testarmos a aplicação vamos vericar o arquivo POM.XML para certificarmos se está correto.

Clique no arquivo e selecione a aba *Dependecy Hierachy*

Veja se há a dependência para o Primefaces-Mobile

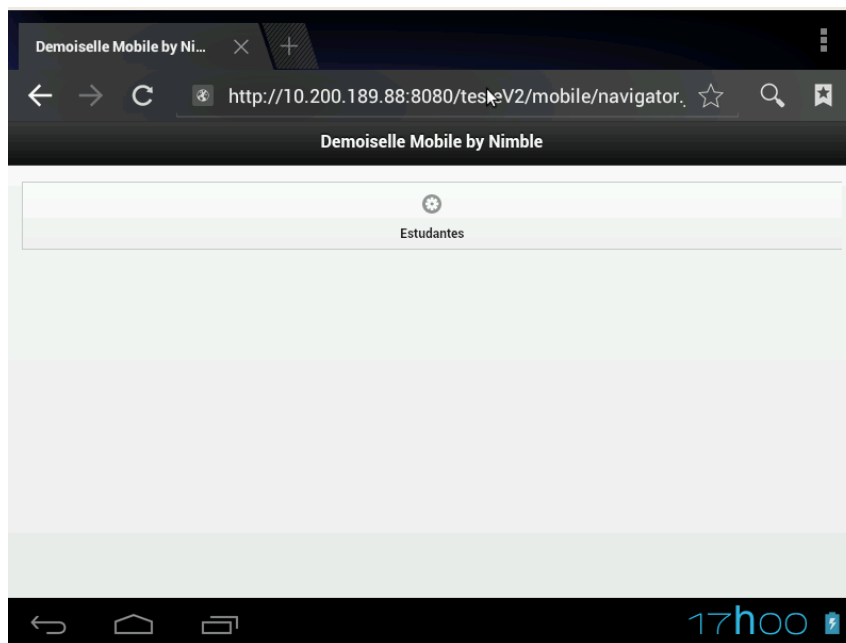
É importante verificar também a versão do Primefaces que deve ser no mínimo a 3.4

Visão de Dependency Hierarchy no POM.XML

Com isso a aplicação já está pronta para ser testada.

Republique novamente no servidor de aplicações e inicie o servidor.

Obviamente não haverá nenhuma diferença ao acessar do navegador (browser) web do seu computador, o ideal é publicar na internet e para isso há alguns serviços em nuvem que fornecem ambiente gratuíto e compatível com o Demoiselle como o *Openshift* [http://openshift.redhat.com/] da RedHat. E depois testar em um tablet ou celular com navegador web.

Uma estratégia que adotamos foi usar o software *VirtualBox* [https://www.virtualbox.org/] e criar uma máquina virtual com *Android* [https://www.buildroid.org/]

Acessando de um navegador em dispositivo móvel a intefaces será assim:

Tela de navegação - Mobile

Tela de listagem

Tela de Edição

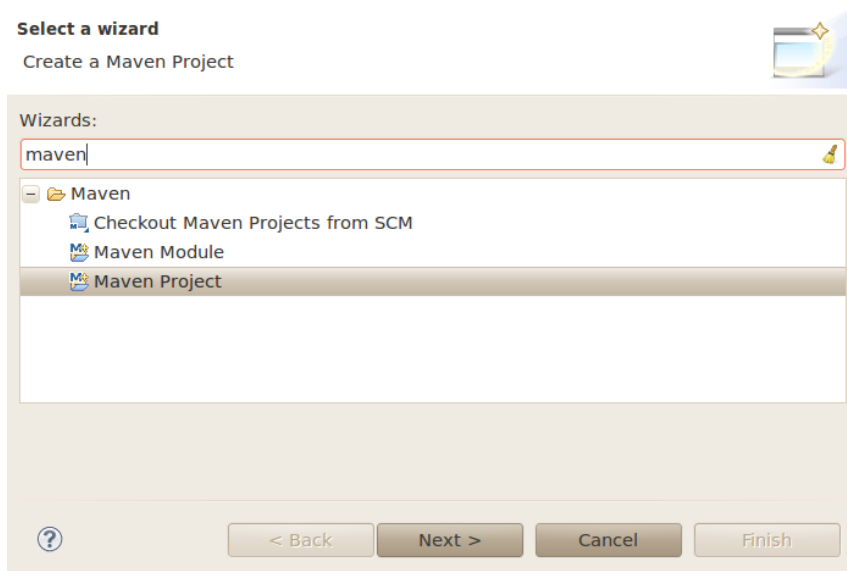
Exemplo de CRUD Vaadin usando Eclipse IDE

Exemplo prático usando o ambiente de Desenvolvimento *Eclipse IDE* para gerar aplicação CRUD Vaadin

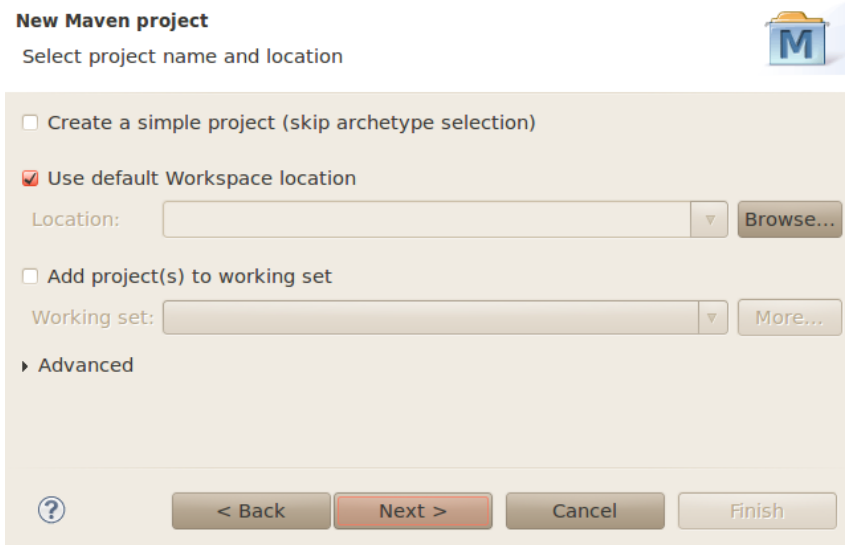
Vamos considerar que a instalação e configuração do Nimble já esteja de acordo com as intruções contidas na *Seção 4.1, “Instalação”* ou preferenciamento que esteja utilizando o *Demoiselle-Infra* [http://demoiselle.sourceforge.net/infra/]

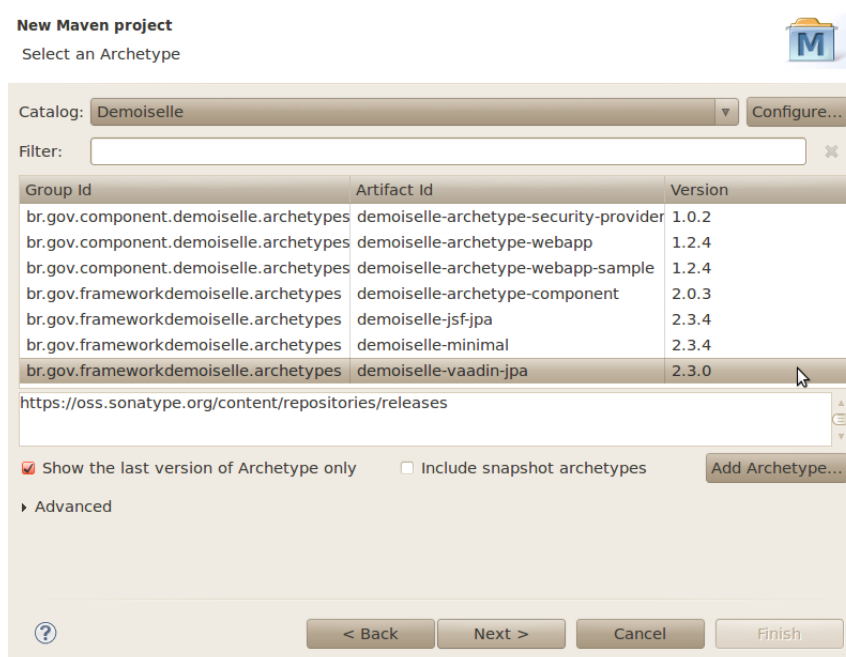
8.1. Prepararação

Antes de iniciar o uso do Demoiselle-Nimble, vamos criar um projeto usando um arquétipo do Demoiselle.

Selecione o Menu do Eclipse *File-> New-> Project...* que irá acionar a seguinte tela

Tela para criação de novos projetos no Eclipse

Digite a palavra *Maven*no campo Wizards para em seguida selecionar a opção Maven Project e depois clique no botão **Next->**

Na tela seguinte mantenha os valores padrão e clique novamente em **Next->**

Definindo workspace para criar o projeto

Na tela seguinte, considerando que esteja usando o Eclipse provido pela instalação do Demoiselle-Infra, com o catálogo *Demoiselle* selecionado escolha o arquétipo *demoiselle-vaadin-jpa* que irá criar um projeto Vaadin utilizando a extensão JPA.

Escolhendo o arquétipo Demoiselle para VAADIN

Na tela seguinte, adicionamos as informações basicas do projeto

Escolhendo o arquétipo Demoiselle para VAADIN

Digite as seguintes informações nos campos:

- *Group Id:* br.org.frameworkdemoiselle
- *Artifact Id:* testeV

Os demais o Eclipse já preenche automaticamente

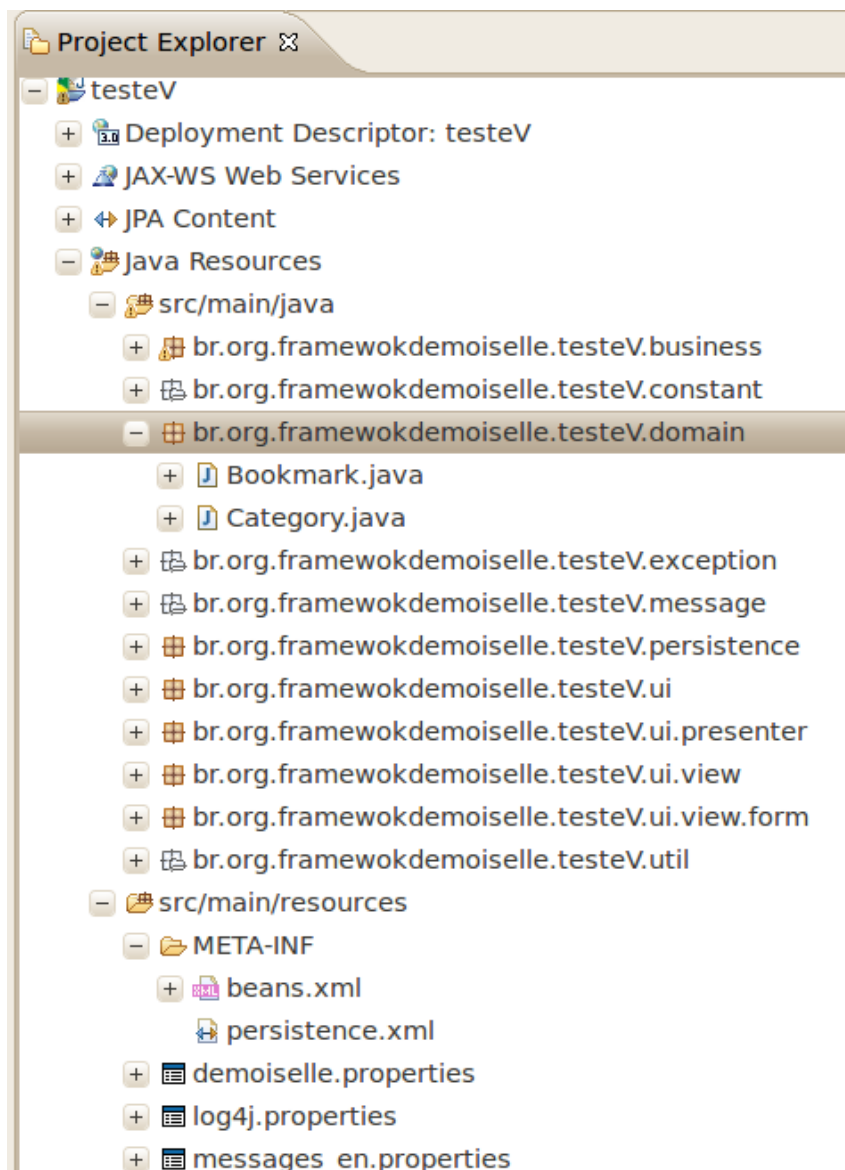
Em seguinda clique no botão *Finish* e aguarde que o processo seja concluído pelo Eclipse.

Verifique o arquivo /src/main/resources/META-INF/persistence.xml e remova os comentários para escolher a estratégia de transação. Conforme mostrado abaixo:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.0" xmlns="http://java.sun.com/xml/ns/persistence" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://java.sun.com/xml/ns/persistence http://java.sun.com/xml/ns/
persistence/persistence_2_0.xsd">

<!-- JBoss6 ou JBoss7 -->
    <persistence-unit name="bookmark-ds" transaction-type="RESOURCE_LOCAL">
        <non-jta-data-source>java:jboss/datasources/ExampleDS</non-jta-data-source>
        <class>br.org.framewokdemoiselle.testeV.domain.Bookmark</class>
        <class>br.org.framewokdemoiselle.testeV.domain.Category</class>
        <properties>
            <property name="javax.persistence.jdbc.driver" value="org.hsqldb.jdbcDriver" />
            <property name="javax.persistence.jdbc.user" value="sa" />
            <property name="javax.persistence.jdbc.password" value="" />
            <property name="hibernate.show_sql" value="true" />
            <property name="hibernate.format_sql" value="false" />
            <property name="hibernate.hbm2ddl.auto" value="create-drop" />
        </properties>
    </persistence-unit>
    </persistence>
```

No arquivo /src/main/resources/messages.properties, encontre e muda a propriedade:

main.app.title=TesteVaadin

Após criado o projeto, vamos localizar o pacote *Domain* conforme a figura abaixo:

Localizando o pacote das classes de domínio

Crie neste pacote uma classe chamada *Pessoa*, conforme o código abaixo

```
package br.org.framewokdemoiselle.testeV.domain;

import java.io.Serializable;
import java.util.Date;
import javax.persistence.*;
import br.gov.frameworkdemoiselle.vaadin.annotation.*;

@Entity
@Table(name="TB_Contato")
public class Contato implements Serializable{
```

```
/**
 *
 */
private static final long serialVersionUID = 1169988670196408411L;

@Id
@GeneratedValue
private Long id;

@Column(nullable=false,  length=255)
@TextField
@Field(prompt = "{Contato.prompt.nome}", label = "{Contato.label.nome}")
private String nome;

@Column
@CpfField
@Field(prompt = "{Contato.prompt.cpf}", label = "{Contato.label.cpf}")
private String cpf;

@Column(nullable=false)
@Temporal(value=TemporalType.DATE)
@Field(prompt = "{Contato.prompt.datanascimento}", label = "{Contato.label.datanascimento}")
@DateField
private Date dataNascimento;

@Column
@PhoneField
@Field(prompt = "{Contato.prompt.telefone}", label = "{Contato.label.telefone}")
private String telefone;

@Column
@RichText
@Field(prompt = "{Contato.prompt.observacoes}", label = "{Contato.label.observacoes}")
private String observacoes;

public Long getId() {
    return id;
}

public void setId(Long id) {
    this.id = id;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}

public String getCpf() {
    return cpf;
}

public void setCpf(String cpf) {
    this.cpf = cpf;
}
```

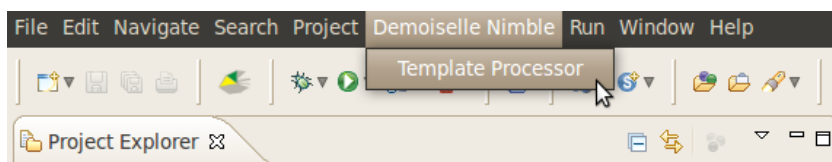
```
        public Date getDataNascimento() {
            return dataNascimento;
        }

        public void setDataNascimento(Date dataNascimento) {
            this.dataNascimento = dataNascimento;
        }

        public String getTelefone() {
            return this.telefone;
        }

        public void setTelefone(String telefone) {
            this.telefone = telefone;
        }

        public String getObservacoes() {
            return this.observacoes;
        }

        public void setObservacoes(String observacoes) {
            this.observacoes = observacoes;
        }

}
```

8.2. Gerando a Aplicação

Ainda no ambiente Eclipse, selecione a pasta principal do projeto e veja onde estão os acionadores do Plugin Demoiselle Nimble

- Um ícone na barra de ferramentas:

Icone do Demoiselle Nimble na barra de ferramentas

- Sub-item no Menu Ferramentas

Sub-item Demoiselle Nimble

Com o projeto selecionado, use uma das opções acima para acionar a interface do Demoiselle Nimble, onde selecionaremos o Template para geração de um CRUD (Create, Read, Update e Delete) que são as operações básicas de criar, ler, atualizar e apagar

Selecione o Template adequado ao arquétipo que foi criado, neste exemplo optamos pelo *Demoiselle V2 Generate Vaadin CRUD*, que é o tipo de aplicação que estamos criando, conforme a figura abaixo:

Escolhendo o template de CRUD

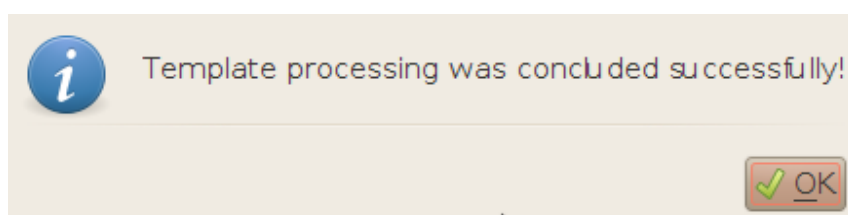
Na tela seguinte, deverão ser informados os parâmetros para geração dos artefatos

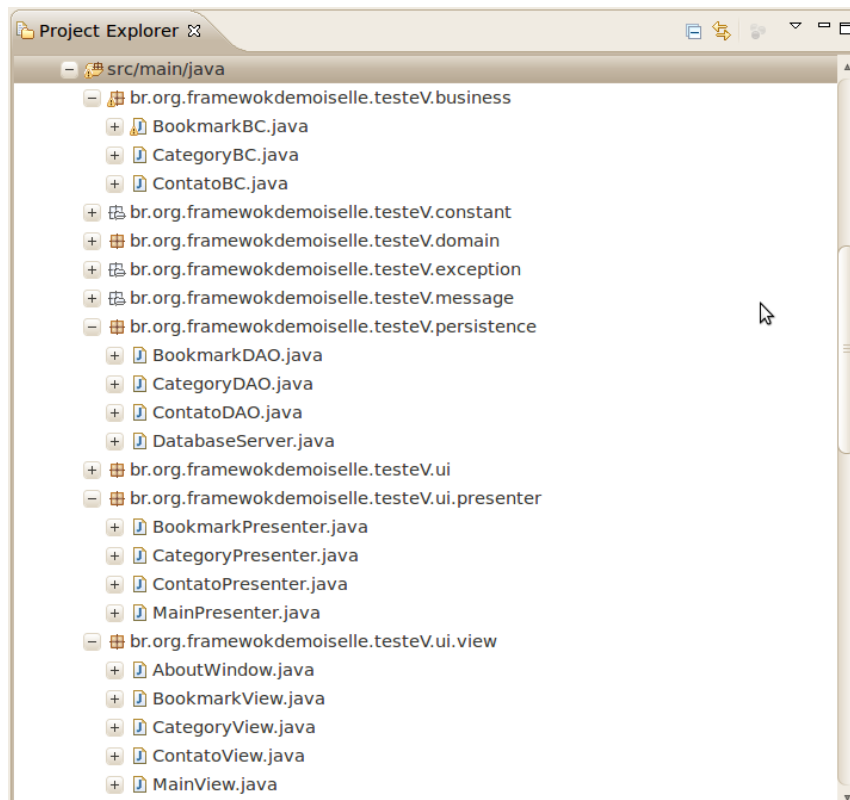
1. Output Folder*: Use o botão *Browse...* para selecionar o projeto dentro do diretório de Workspace do Eclipse, caso ainda não esteja selecionado

2. Entity Name*: Use o botão *Click to select an entity class file...* e procure no diretório (/src/main/java/br/org/frameworkdemoiselle/testeV/domain/) a classe Contato

3. Package Name*: Será preenchido automáticamente com *br.org.frameworkdemoiselle.testeV*

Informando os parâmetros para criação dos artefatos

Aguarde o processamento até que a tela de confirmação apareceça

Processamento do Template OK

Confira os artefatos que foram gerados nos pacotes de negócio(business), persistência (persistence) e de visão (presenter e view)

Listagem dos artefatos criados

Veja também nos arquivos: persistence.xml e messages.properties, que novas informações foram incluídas.

Assim como os arquivos MainPresenter.java e MainView.java que também foram editados

Para testar as funcionalidade criadas, adicione o projeto no servidor JBoss e inicie o mesmo.

No exemplo a aplicação estará em http://localhost:8080/testeV

Veja algumas telas abaixo:

Resultado na aplicação em funcionamento

Resultado na aplicação em funcionamento

Resultado na aplicação em funcionamento